An Interactive and Iterative Knowledge Extraction Process Using Formal Concept Analysis

My Thao Tang

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Un système interactif et itératif extraction de connaissances exploitant l'analyse formelle de concepts

THÈSE

présentée et soutenue publiquement le 2016

pour l'obtention du

Doctorat de l'Université de Lorraine

(spécialité informatique)

par

My Thao TANG

Composition du jury

Rapporteurs :	Henri SOLDANO	Maitre de conférence, HdR, Université Paris-Nord
	Anne VILNAT	Professeur, Université Paris Sud
Examinateurs :	Amedeo NAPOLI	Directeur de recherche, CNRS, Nancy
	Jean-Marie PIERREL	Professeur, Université de Lorraine
	Nathalie PERNELLE	Maitre de conférence, Université de Paris Sud
Directeurs de thèse :	Yannick TOUSSAINT	Chargé de Recherche, HdR, INRIA Nancy

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Table of contents

Chapter 3
KESAM: A Tool for Knowledge Extraction and Semantic Annotation Management

Chapter 4
Formal Knowledge Structures and "Real-World"

Chapter 5
Conclusion and Perspectives

List of Tables

List of Figures

Chapter 1

Introduction

Contents

1.1 Context and Motivation

Semantic annotation and knowledge extraction for building ontologies are two key processes of knowledge-based information systems. Anyone would agree that these two processes are at least dual, i.e. they should be and could be intimately related or, at most, one two-side but a unique process. However, because of the complexity of the process, it is almost always treated as two separated tasks. This thesis unifies these two processes into one and relies on Formal Concept Analysis [Ganter and Wille, 1999]. In what follows, we introduce semantic annotation and knowledge extraction, then the problems that motivate the work in the thesis.

On the one side, semantic annotation is a helpful solution to enhance information retrieval and improve interoperability [Uren et al., 2006, Bontcheva and Cunningham, 2011, Liao et al., 2015]. Information retrieval is enhanced by the ability of performing searches, which exploit the ontology to make inferences about the data. For example, a search engine can answer for a query of "city" with a text containing "Paris" thanks to a semantic annotation linking "Paris" in that text to an object of concept "city" in an ontology. Interoperability is improved by the ability of understanding by both humans and machines. With semantic annotations, documents can be processed in an intelligent way and efficiently exploited by machines for automation purpose. An annotation is the content represented in a formal language and attached to the document [Euzenat, 2002]. From a technological perspective, semantic annotation is about identifying in text all mentions of concepts from the ontology (i.e. concepts (or classes), objects (or instances), attributes (or properties), and relations) [Bontcheva and Cunningham, 2011]. In an ontology, a concept can be *atomic* or *defined* by some definitions. For example, a concept "car" can be defined by "a road vehicle, typically with four wheels, powered by an internal-combustion engine and able to carry a small number of people" [1] One problem with semantic annotation is to manage of getting annotations that are mentions of *defined concepts*. The first generation of semantic annotation tools was mainly technological and terminological, focusing

[1] `http://www.oxforddictionaries.com/definition/english/car`

on terms and named entity recognition by mapping terms or named entities found in texts to existing instances of concepts in an ontology. These techniques may be not able to identify these defined concepts in texts and then there is a need to take into account defined concepts in semantic annotation. Another problem with semantic annotation is, if an ontology changes, then all or some of the semantic annotations need to be updated to reflex the knowledge updates [Bontcheva and Cunningham, 2011]. These updates are time-consuming and expensive. Semantic annotations need to be co-evolved with ontologies.

On the other side, ontology is the heart of knowledge-based information systems to clarify the knowledge structure of a domain and enable knowledge sharing [Chandrasekaran et al., 1999]. Ontology is defined as "a formal, explicit specification of a shared conceptualization" [Gruber, 1993, Studer et al., 1998]. *Conceptualization* means an abstract model of some aspects of the world, taking the form of a definition of important concepts and relationships. *Formal* means that the conceptualization should be expressed in a (formal) machine readable format. *Explicit* means that the ontology concepts are explicitly defined. *Shared* means that the conceptualization should capture the consensual knowledge, that is, it is not private to some individuals, but accepted by a group or community. Knowledge extraction for building an ontology from texts is a complex [Cimiano, 2006, Aussenac-Gilles et al., 2008, Szulman et al., 2010], time-consuming and costly task. It involves such steps as collecting resources, preprocessing and processing these resources to switch progressively from the linguistic level (words, terms...) to the conceptual level. Domain experts guide this iterative process by favoring the emergence of certain words, concepts, or by introducing nuggets of knowledge which are missing. Most of the works splits building ontology from text in several subtasks leading to a loss of traceability between the source of knowledge, i.e. the texts, and the ontology. When experts are asked to evaluate the knowledge model or the ontology needs to be maintained or upgraded, experts may need to look back to the texts to understand why the concepts are built. Can we manage to know which linguistic expressions in the texts are used to build a concept? Ontology provides the domain vocabulary used for semantic annotations and semantic annotations can help here to provide the traceability between the knowledge model and the resource, i.e. all mentions of concepts in the texts. Moreover, ontology or the conceptualization needs to be frequently updated to adapt to new knowledge resources, including semantic annotations, new user needs... [Zablith et al., 2013] as semantic annotations need to be updated to reflex the new knowledge. Ontologies and semantic annotations need to be co-evolved. Therefore, it is necessary to develop methodologies for unifying semantic annotation and knowledge extraction in one single process to benefit both semantic annotation and knowledge extraction. Semantic annotations are used to keep the traceability between the knowledge model and its linguistic expression in the texts and the knowledge model is used as the input for improving semantic annotations; any changes in semantic annotations reflexes in the knowledge model and vice versa.

The first objective of this thesis is to propose an interactive and iterative process for knowledge extraction, in which we focus on keeping the co-evolution and the traceability between the knowledge model and the source of knowledge in texts, i.e. semantic annotations. The traceability here means the ability of getting semantic annotations that are used to build concepts in the knowledge model including atomic and defined concepts. With this traceability, domain experts can get explanations why the concepts are built and therefore, it makes easier the evaluation or maintenance of the knowledge model. To achieve these goals, we choose Formal Concept Analysis [Ganter and Wille, 1999] as the data mining method for building the knowledge model, i.e. the concept hierarchy. Formal Concept Analysis (FCA) is a formal method of classification which has proved very efficient as a bottom-up conceptualization method for building ontologies from texts [Fennouh et al., 2014, Poelmans et al., 2013, Jia et al., 2009, Bendaoud et al., 2008c, Ben-

daoud et al., 2008a, Cimiano et al., 2005, Obitko et al., 2004]. In order to overcome the problem of the knowledge acquisition bottleneck, different methods have been proposed for automatically building a concept hierarchy from texts. These methods can be grouped into two groups: the *similarity*-based methods and the *set-theoretical*-based methods [Cimiano et al., 2005]. FCA, a method based on order theory, perfectly fit our goals. FCA discovers concepts and order them hierarchically at the same time. It provides a good traceability in a comparison with divisive and agglomerative clusterings [Cimiano et al., 2004b, Cimiano et al., 2005]. Moreover, we would like a way to take into account both atomic and defined concepts, not only concepts with names and nothing inside; FCA provides intensional and extensional descriptions for concepts, therefore providing better understanding to users [Cimiano et al., 2004b, Cimiano et al., 2005] and allowing us to take into account both atomic and defined concepts.

The second objective that we would like to achieve in the thesis is to assist human and machine in refining the knowledge model based on the concept lattice resulting from FCA, and semantic annotations. Knowledge extraction is an iterative and interactive process involving several steps. In this process, interaction is usually described as evaluation, where domain experts are invited to interpret and validate the patterns extracted by data mining algorithms. However, any formal method has weaknesses when modeling a cognitive process which is not governed by clear and precise rules. Domain experts may understand the domain in a different way than what is represented in data, and often there exists a gap between the knowledge model based on a concept lattice and the knowledge model of a domain expert. For example, in the domain of animals, an expert may expect that the rule "mammal implies do not lay eggs" holds, while this may not be the case if the platypus is among the objects in the formal context. Domain experts may not be happy with the knowledge model provided by the concept lattice and thus, would like to make it to be more in accordance with their own knowledge or applications' need. There are several reasons that can lead to the gap between the knowledge model based on a concept lattice and the knowledge model of a domain expert: (1) the fact that FCA builds formal concepts from object descriptions (bottom-up approach) makes it prone to unwanted concept creation if there is noise, errors or exceptions in the data; (2) formal concepts may represent unwanted levels of granularity. (3) the knowledge model based on the concept lattice is not in accordance with expert knowledge; (4) experts wish the concept lattice to be more in accordance with their needs, i.e. the tasks or applications that will use the knowledge model. Thus, experts should be able to correct or to refine the knowledge model based on the concept lattice. In a return, machine should be able to take into account the expert actions to improve the knowledge model and semantic annotations. An approach should be proposed to support the formal method, i.e. FCA, can live together with expert interaction.

1.2 Approaches and Contributions of the Thesis

To meet the problems presented above, we propose in this thesis a methodology using Formal Concept Analysis for knowledge extraction and semantic annotation management (KESAM).

The first contribution of this thesis is to build an interactive and iterative process for knowledge extraction from texts based on Formal Concept Analysis [Ganter and Wille, 1999] that is able to keep the co-evolution and the traceability between the knowledge model and semantic annotations of texts. In our approach, knowledge extraction and semantic annotation are unified into one single process to benefit both knowledge extraction and semantic annotation. Formal Concept Analysis [Ganter and Wille, 1999] is the core of the KESAM process for building the knowledge model based on the concept lattice and ensuring the link between semantic annota-

tions and each knowledge units in the knowledge model. In order to get the concept lattice as close as possible to domain experts' requirements, the KESAM process enables expert interaction by introducing the sets of changes in the formal context, the concept lattice and semantic annotations. Domain experts can evaluate and make changes in the formal context, the concept lattice or semantic annotations until they reach an agreement between the knowledge model and their own knowledge or requirements. Along with a change, the consequences of that change in the lattice is reported to domain experts so that domain experts can decide if that change should be applied. The incremental lattice building approach is used for implementing the changes and managing the consequences of the changes in the lattice. The KESAM process then is in charge of keeping the formal context, the concept lattice and semantic annotations up to date with the changes. In such an interactive and iterative process, the system is able to keep the consistency between the knowledge model and semantic annotations, and converges towards a knowledge model close to the requirements of domain experts. Thanks to the link between the knowledge model and semantic annotations, the traceability between the knowledge model and semantic annotations can be enabled; semantic annotations and the knowledge model can co-evolve. Moreover, FCA allows to build definitions of concepts with a sets of objects and sets of attributes and therefore, this makes possible the annotation of both atomic and defined concepts.

Our second contribution is a formal method for bridging the possible gap between the knowledge model based on a concept lattice and the knowledge model of a domain expert. In order to get the concept lattice as close as possible to the experts' requirements for building a knowledge base, domain experts are invited to provide their knowledge as a set of attribute dependencies [Belohlavek and Sklenar, 2005] or constraints which is "aligned" with the set of implications provided by the concept lattice, leading to modifications in the original concept lattice. The method can be generalized for generating lattices guided by constraints based on attribute dependencies. This method also allows the experts to keep a trace of the changes occurring in the original lattice and the revised version, and to assess how concepts in practice are related to concepts automatically issued from data. We are able to build the constrained lattices and provide the trace of changes by using extensional projections [Ganter and Kuznetsov, 2001, Pernelle et al., 2002] over lattices. From an original lattice, two different projections produce two different constrained lattices, and thus, the gap between the knowledge model based on a concept lattice and the knowledge model of a domain expert is filled with projections without changing the original data.

The third contribution of the thesis consists of the system implementation, the experiments and the evaluation. The proposed approaches have been implemented in the KESAM system. We experimented the approaches for building knowledge bases about rare diseases. The approaches are independent of application domains and can be applied in any field. Another experiment for building knowledge bases about animals is also presented in the thesis. The results of the experiments show the capabilities of the approaches.

1.3 Thesis Organization

The thesis is organized into five chapters as follows. The first chapter presents a general introduction. The second chapter introduces some basic concepts, terminologies, the state of the art related to the problems of this thesis and sets the context of the thesis. The two following chapters detail the contributions of the thesis. The final chapter draws a conclusion and some perspectives.

Chapter 2: Knowledge Extraction and Semantic Annotation of Texts. First, this chapter provides the background knowledge on knowledge extraction, semantic annotation, and the problems that we aim to solve in this thesis. Then, the theoretical foundations of Formal Concept Analysis that used through the thesis are introduced. Next, we survey the related work on knowledge extraction and semantic annotation with FCA, especially analyze their advantages and disadvantages related to our proposal. Finally, we conclude all elements help us to achieve our objectives.

Chapter 3: KESAM: A Tool for Knowledge Extraction and Semantic Annotation Management. In this chapter, we present our methodology and the system named KESAM for Knowledge Extraction and Semantic Annotation Management. The KESAM system is used for building a knowledge model from semantic annotated texts. The novelty of the KESAM system is that, it provides the traceability between the knowledge model and semantic annotations including both atomic and defined concepts and keeps the co-evolution between the knowledge model and semantic annotations. Formal Concept Analysis is placed at the center of the system to build the knowledge model based on the concept lattice iteratively and ensure the link between the knowledge model and semantic annotations. Thanks to this link, both the knowledge model and semantic annotations can be efficiently updated. In addition, by classifying concepts according to their attributes using FCA, the link can be preserved for both atomic and defined concept annotation. At the end of the chapter, we present the implementation of the KESAM system and a case study on medical domain. The approach was applied to build knowledge bases about rare diseases with sets of abstracts from PubMed[2]. The results of applying the approach for building and refining the classification of diseases are presented and analyzed in details.

Chapter 4: Formal Knowledge Structures and "Real-World". In this chapter, we present our formal method for bridging the possible gap between the knowledge model based on a concept lattice and the knowledge model of a domain expert. In this work, in order to bridge the gap above, we "align" a set of attribute dependencies with the set of implications provided by the concept lattice, leading to modifications in the original lattice. The method extends the definition of dependencies between single attributes introduced in [Belohlavek and Sklenar, 2005] to the case of dependencies between attribute sets, and allows domain experts to have more possibilities for expressing constraints. We then define the extensional projection for constraining the concept lattice w.r.t. sets of dependencies between attribute sets and providing the trace of changes occurring in the original lattice and the revised version. The order of projections for generating a constrained lattice w.r.t. a set of dependencies is examined in detail.

Chapter 5: Conclusion and Future Work. This chapter concludes our work and discusses about possible perspectives for the study.

[2] `http://www.ncbi.nlm.nih.gov/pubmed`

Chapter 2

Knowledge Extraction and Semantic Annotation of Texts

Contents

2.1 Introduction

This chapter is to provide the readers the background knowledge, the context and the rationale behind our work. In this chapter, we first present the fundamentals of knowledge extraction and semantic annotation of texts that supply for better understanding our work. Then, we introduce the theoretical foundations of Formal Concept Analysis (FCA), the classification method that we choose for our knowledge extraction process. Next, we present a survey of the work on knowledge extraction and semantic annotation with FCA related to our work, their strategies, as well as

their advantages and disadvantages to support our research. Finally, we conclude all elements help us to achieve the objectives of the thesis.

2.2 Knowledge Extraction

In this section, we first present the knowledge extraction process from databases. Then, the adaptation of the knowledge extraction process from databases to textual resources is described, followed by a discussion on the challenges of this process. Next, we review some basic concepts, terminologies and characteristics of an ontology that support to build an interactive and iterative process for knowledge extraction from texts. After that, we discuss about the classification methods for building the ontology backbone and choose the classification method for our process. Finally, we review the related work on interactive knowledge extraction from texts.

2.2.1 Knowledge Extraction Process

The knowledge discovery in databases (KDD) process in general was defined by Fayyad et al. [Fayyad et al., 1996] and then detailed in several books [Brachman and Anand, 1996, Dunham, 2002] as a process of transforming low-level data into a more compact, abstract, and useful form. The KDD process aims at extracting knowledge units that are non trivial, potentially useful, significant, and reusable, and can be seen as a process of turning data into information and then knowledge [Napoli, 2005]. This process is interactive and iterative, and guided by experts of the data domain in order to extract knowledge units on the base of their requirements or their own knowledge [Napoli, 2005]. It is iterative because it can run several times in order to obtain the requirements of the domain experts.

The KDD process is performed within a KDD system that consists of the following elements:

- the databases,
- the data mining methods,
- the interfaces for interactions with the system.

The KDD process consists of four main stages: (1) first, the data sources are selected and collected, (2) then, the collected data are prepared, (3) next, data mining methods are applied to the prepared data to extract interesting information units, and (4) finally, the extracted information units are evaluated by domain experts for becoming knowledge units. The knowledge units extracted by the KDD system are then used to build a model of the domain and stored in a knowledge base, a domain ontology to be reused for problem solving needs in application domains. The model of the domain will be regarded as *the knowledge model* of an ontology. More precisely, the main steps of the KDD process (see Figure 2.1) are:

- **Data selection:** In this step, the working data sets are selected and collected.
- **Data preparation:** This step consists of cleaning the collected data and then transforming them into a suitable format for the selected data mining method.
- **Data mining** This step deals with extracting information units (*discovered patterns*) from the prepared data.
- **Interpretation/Evaluation:** In this step, the extracted information units are interpreted and evaluated by domain experts for becoming knowledge units.

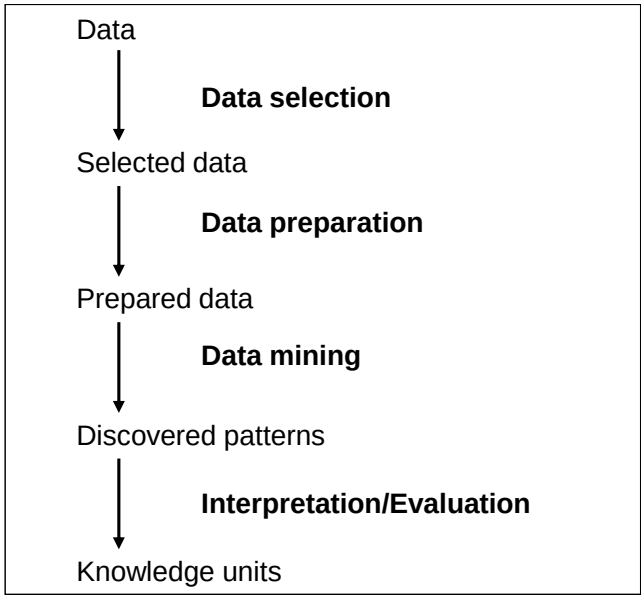

Figure 2.1: The KDD process. Figure adapted from [Napoli, 2005].

First, in the data selection step, according to the objectives of the process, the domain, such as biology, chemistry, or medicine..., and the resources are determined to collect the data. The data can be collected from many various resources such as databases, files, texts...

Next, in the data preparation step, the collected data are cleaned by removing noise, inconsistencies..., and then, the data are combined into a common source and transformed into a suitable format for the selected data mining methods.

After the data preparation step, in the data mining step, data mining methods are applied to extract information units of the prepared data. The data mining methods for KDD systems can be either *symbolic* or *numerical* [Napoli, 2005]. Symbolic methods include classification based on decision trees, lattice-based classification, frequent itemsets search, association rule extraction, classification based on rough sets [Pawlak, 1992], instance-based learning, explanation-based learning [Mitchell, 1997, Michalski et al., 1998]... Numerical methods include statistics and data analysis, hidden Markov models, neural networks [Ghahramani, 2002]... The choice of the data mining method depends on the purpose of a KDD system such as classification, categorization, data summarization, or data model creation... Among symbolic data mining methods, lattice-based classification [Barbut and Monjardet, 1970, Guénoche and Mechelen, 1993, Ganter and Wille, 1999] is an efficient method that can be used for extracting concepts from data and organizing them within a concept hierarchy. In this thesis, we choose the lattice-based classification method, Formal Concept Analysis (FCA) [Ganter and Wille, 1999], as the data mining method for our knowledge extraction process. In the next chapter, we will show the process of transforming rough data into knowledge units using FCA.

After the data mining step, the extracted information is represented to domain experts for interpretation and evaluation through visualization techniques for becoming knowledge units. Domain experts play an important role in the knowledge extraction process. They have to decide what is considered knowledge according to their requirements or their own knowledge. This task is very difficult to achieve due to several reasons such as: (1) the information units extracted by data mining methods are not always easy to understand for domain experts; (2) some

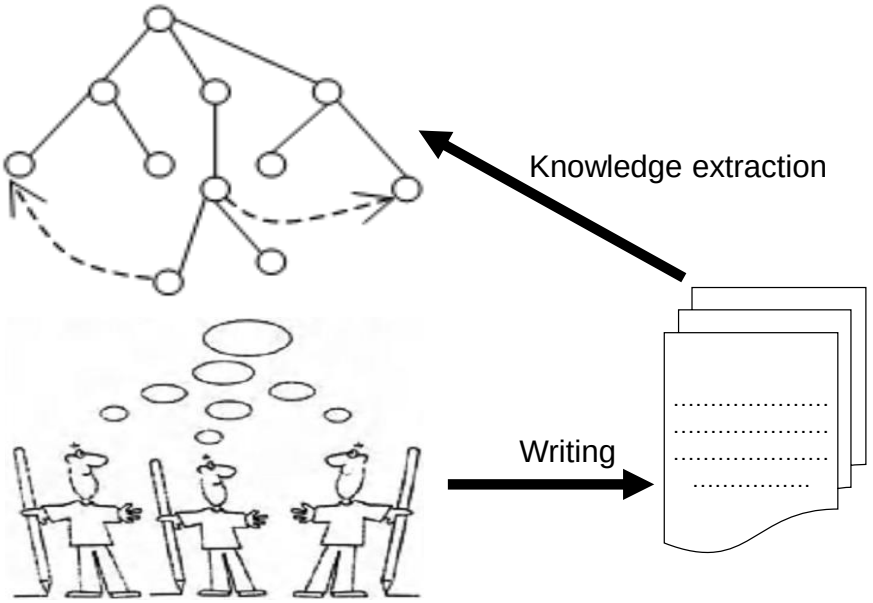

Figure 2.2: Knowledge extraction from texts. Figure adapted from [Cimiano, 2006].

background knowledge is often assumed or expressed implicitly in the data, especially in textual resources; (3) the data collections are never big enough to provide all the knowledge; (4) the extracted information units are not in accordance with experts' knowledge or their requirements for the tasks or applications. Particularly, often there exists gap between a knowledge model resulting from a bottom-up approach and a model that domain experts expect because humans often follow top-down approaches when building knowledge bases. For example, in a knowledge model resulting from a bottom-up approach, a `chicken` and a `penguin` can be grouped together into a concept because these objects share a common attribute `has wings`, this extracted information may not be in accordance with expert knowledge because the attribute `has wings` is not meaningful enough for them to characterize the objects `chicken` and `penguin`. Thus, the knowledge extraction system should be able to support domain experts in interpreting, analyzing, and refining the information units resulting from the data mining methods.

In this thesis, we focus on the formalization of expert knowledge as well as the discovery of knowledge units from textual resources. In the next sub-section, we discuss about the challenges of the knowledge extraction process from texts as it has some particular characteristics.

2.2.2 Knowledge Extraction from Texts

In case the KDD process is performed on textual resources, it is called *knowledge extraction from texts*. A system for knowledge extraction from texts consists of the three main following elements: the first element is the text datasets, the second element is the data mining methods, and the third element is the interfaces for interpretation and evaluation by domain experts. In 2001, Maedche and Staab [Maedche and Staab, 2001] described learning ontology from texts as an acquisition process of a domain model from textual data. In 2006, Cimiano [Cimiano, 2006] considered the construction of an ontology from texts as a reverse engineering process. According to Cimiano [Cimiano, 2006], the authors of a certain text or document have a domain model in mind when they are writing it. The process of knowledge extraction from texts thus can be

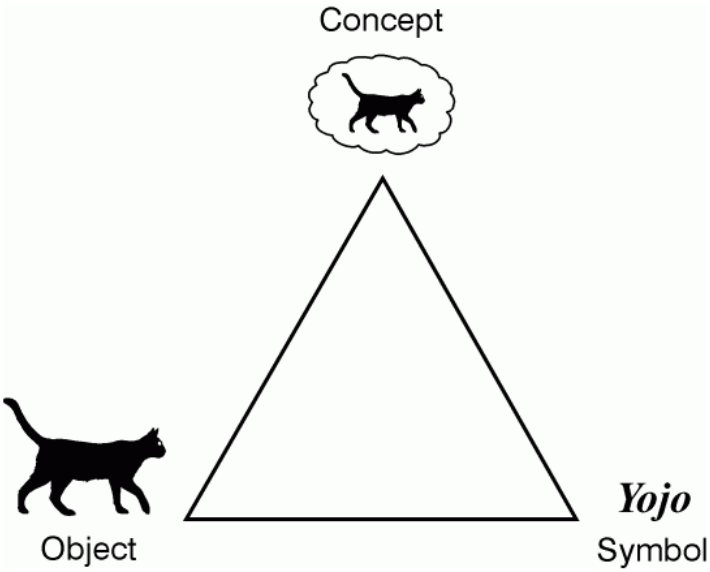

Figure 2.3: Example of a meaning triangle. Figure taken from [Sowa, 2000].

considered as a reverse engineering process, in which we should try our best to reconstruct the authors' model (see Figure 2.2).

Knowledge extraction from texts is complex and challenging. As Cimiano [Cimiano, 2006] claimed, the reverse engineering process only can partially reconstruct the authors' model because there is a difference between what is in the authors' mind and what they describe in texts. Moreover, it often requires some background knowledge to understand texts properly because background domain knowledge is often assumed or mentioned implicitly in the texts [Brewster et al., 2003]. Figure 2.3 shows an example of a meaning triangle in dealing with the meaning of texts [Sowa, 2000]. In this example, there is an object, a cat named Yojo, on the left corner. On the right corner, there is a symbol that represents her name, Yojo. And, on the top corner, there is a concept that relates the symbol to its object, i.e. the model in mind of an author. When some authors want to talk about the cat Yojo (the object), they may use the symbol Yojo to describe it in texts. To reconstruct the authors' model, the knowledge extraction system or the readers should understand that the symbol Yojo in the texts is mentioning about the cat Yojo; this information is not mentioned explicitly in the texts. The role of domain experts in the process of knowledge extraction from texts is thus more important, and their task in this process is more difficult to achieve because the information units extracted automatically need to be interpreted, validated and refined for becoming knowledge units before they can be used for problem solving needs in applications.

This thesis is dedicated to domain experts for extracting knowledge from texts. We aim at building an iterative and interactive knowledge extraction process, where domain experts are assisted in interpreting, evaluating, and refining the knowledge model. Knowledge extraction from texts is a complex, time-consuming and costly task. Splitting knowledge extraction from texts into several tasks could lead to lose the *traceability* between the knowledge model and the source of knowledge in texts The traceability here means, for each knowledge unit in the knowledge model, i.e. a concept, experts can trace back to the source of knowledge in texts to understand why a concept is introduced in the knowledge model. For example, from a concept containing the objects `chicken` and `penguin` and the attribute `has wings` in the knowledge

model, experts can trace back to texts to see where we get the information, a `chicken has wings` and a `penguin has wings`. Losing the traceability between the knowledge model and the source of knowledge in texts might cause difficulties for domain experts in the interpretation, evaluation, and refinement of the knowledge model [Cimiano and Völker, 2005, Aussenac-Gilles et al., 2008, Szulman et al., 2010]. Our first objective in this thesis is to provide domain experts the traceability between the knowledge model and the source of knowledge in texts to support domain experts in interpretation, evaluation, and refinement of the knowledge model. By tracing back to the source of knowledge in texts, experts can have some explanations why concepts in the knowledge model are built and therefore, it makes easier the interpretation, evaluation, and refinement of the knowledge model.

Our second objective in this thesis is to assist domain experts in refining the knowledge model produced automatically by the data mining method, i.e. Formal Concept Analysis, so that domain experts can get the knowledge model as close as possible to their requirements through an interactive process. Refining the knowledge model means, domain experts can make changes in the knowledge model, add some background knowledge to the source of knowledge or add constraints to get a better knowledge model with respect to the experts' requirements. For example, if experts don't want a concept that groups a `chicken` and a `penguin` together, they can ask for removing this concept from the knowledge model to get the knowledge model to be more in accordance with their requirements. An example of adding background knowledge is, with the texts using only the name `Yojo` for mentioning about the cat `Yojo`, experts can add "Yojo is a cat" to the source of knowledge to get the knowledge model more comprehensive. An example of adding constraints is, an expert may expect that the rule "mammal implies do not lay eggs" holds, while this may not be the case if the platypus is among the objects in the context. The system then should take into account the requirements from domain experts in the knowledge model and the source of knowledge. Moreover, modifications in the knowledge model may lead to update in the source of knowledge, and conversely [Maedche and Staab, 2001, Bontcheva and Cunningham, 2011]. These updates are expensive and time consuming. Our objective here is twofold, besides the objective of assisting domain experts in refining the knowledge model, we also aim at keeping the co-evolution of the knowledge model and the source of knowledge, i.e. any changes in the knowledge model reflexes in the source of knowledge, and vice versa.

In what follows, we review some basic concepts and characteristics of ontologies that support to build our knowledge extraction process.

2.2.3 Ontology

Computational ontologies are a means to formally model the structure of a knowledge-based system and enable knowledge sharing. As we have seen in section 2.2.1, the knowledge units extracted from the knowledge extraction system are then stored in an ontology to be reused for problem solving needs. In computer science, ontology was firstly defined as "an explicit specification of a conceptualization" by Gruber [Gruber, 1993]. In this definition, *conceptualization* means an abstract model of important concepts in a domain of discourse and relationships between those concepts, and *explicit* means that the conceptualization is explicitly defined. Later, in 1997, Borst [Borst, 1997] defined an ontology as a "formal specification of a shared conceptualization". This definition requires the conceptualization must be expressed in a (*formal*) format that computers can read, such as Description Logic (DL) [Baader et al., 2003], Web Ontology Language (OWL)[3], etc., and captures *sharing* knowledge that is not only accepted by an indi-

[3] `http://www.w3.org/TR/owl-ref/`

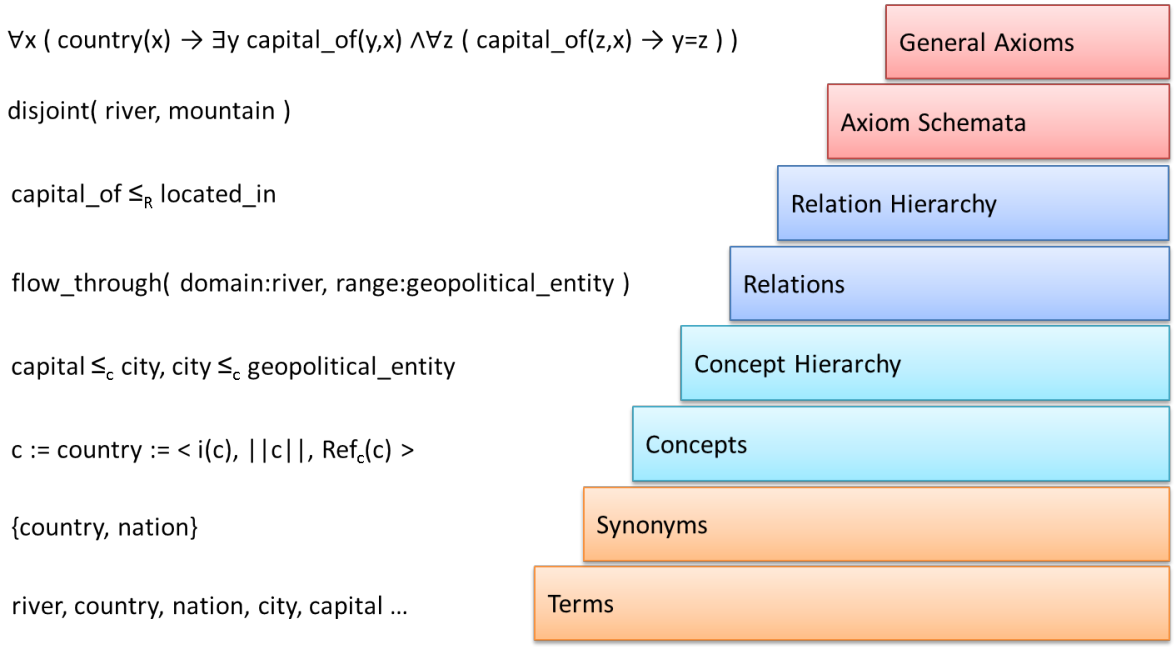

Figure 2.4: Ontology learning layer cake. Figure taken from [Cimiano, 2006].

vidual, but also a group or a community. In 1998, these two definitions [Gruber, 1993, Borst, 1997] were merged into one as "an ontology is a formal, explicit specification of a shared conceptualization" by Studer et al. [Studer et al., 1998]. In 2009, Guarino et al. [Guarino et al., 2009] revisited, discussed in details the definition from Studer et al. [Studer et al., 1998] and claimed that ontologies need to fulfill the communication between human and machine.

Concepts (or *classes*) are the main elements of an ontology. For example, a concept of animals represents all animals. Specific animals such as `chicken` and `penguin` are *objects* (or *instances*, or *entities*) of this concept. The backbone of an ontology consists of a generalization/specification hierarchy of concepts [Guarino et al., 2009].

Ontology learning from texts is about identifying terms, concepts, relations, and optionally axioms from texts to construct ontologies. The output of ontology learning from texts is described as the layers in an *ontology learning layer cake* [Buitelaar et al., 2005, Cimiano, 2006] (see Figure 2.4). The tasks of ontology learning from texts corresponding to the layers of the ontology learning layer cake are [Cimiano, 2006]:

- acquisition of the relevant terminology,
- identification of synonym terms/linguistic variants,
- formation of concepts,
- hierarchical organization of the concepts (concept hierarchy),
- learning relation, attributes or properties, together with the appropriate domain and range,
- hierarchical organization of the relation (relation hierarchy),
- instantiation of axiom schemata,

- definition of arbitrary axioms.

These tasks can be grouped together and performed at the same time.

There are two kinds of ontologies: *lightweight ontologies* [Davies, 2010] and *heavyweight ontologies* [Furst and Trichet, 2006]. A lightweight ontology is simply based on a hierarchy of concepts and a hierarchy of relations, uses very little or does not use axioms while a heavyweight ontology uses axioms for specification. Lightweight ontologies are the most common kind of ontologies and sufficient for many kinds of applications [Wong et al., 2012]. Wong et al. claimed that ontology learning from texts, in fact, is to create lightweight ontologies [Wong et al., 2012]. In this thesis, we limit ourselves to create lightweight ontologies. In this context, we focus on building the ontology backbone, i.e. the concept hierarchy. We call it *the knowledge model* of an ontology.

In the following, we introduce the definition of terms based on [Cimiano, 2006], and the definitions of concepts and concept hierarchies for the process of knowledge extraction from texts that we aim to build.

Definition 2.1 (Term)**.** *Terms* are linguistic realizations of concepts in a corpus. Terms are either single words or multi-word compounds with a very specific, possible technical meaning in a given context or domain.

In order to provide the traceability between the knowledge model and source of knowledge in texts, we provide domain experts with terms of concepts when experts trace back from concepts in the knowledge model to the source of knowledge in texts.

We distinguish two types of concepts in an ontology: *atomic concepts* (or *primitive concepts*) and *defined concepts* [Baader et al., 2003]. Atomic concepts are simply denoted by names such as `Person, Chair`... They have to be considered as atoms of an ontology and are used for building more complex definitions. The other concepts, i.e. defined concepts, are built by using *definition statements*. A definition statement consists of a set of necessary and sufficient conditions for an individual being an object of a concept. For example, a concept `Mammal` can be defined by a set of attributes, `being warm blooded` and `having four legs`, then every object of `Mammal` `is warm blooded` and `has four legs`. Reciprocally, every individual having attributes `being warm blooded` and `having four legs` is recognized as an object of the concept `Mammal`.

Regarding the objective of providing the traceability between the knowledge model and the source of knowledge in texts, we would like to take into account all the concepts in the knowledge model, both atomic and defined concepts. Therefore, we would like to build concepts with definitions. We use the definition of concepts based on [Buitelaar et al., 2005, Cimiano, 2006] that is suitable to our objective as below.

Definition 2.2 (Concept)**.** A *concept* is a triple $< \Im, \Sigma, T >$, where $\Im$ is the intension of the concept, Σ its extension, i.e. a set of objects (or instances, or entities), and T its terms in a corpus.

There is no explicit definition of an intension. An intension of a concept can be a natural language description of the intuitive meaning of a concept or a set of attributes [Cimiano, 2006]. In the line with the theory of Formal Concept Analysis, we consider an intension of a concept is a set of attributes. For example, an intension of concept `Mammal` can be a set of attributes {`being warm blooded`, `having four legs`}.

The backbone of an ontology, the concept hierarchy structures concepts according to the attributes associated with them. The hierarchical structure is defined in such a way that more

specific concepts inherit the attributes of the more general ones. In data analysis, a hierarchy was defined by Diday [Diday, 1982] as a tree which has no multiple inheritances. In the area of knowledge extraction and FCA, a hierarchy is no longer considered as a tree, but a partial order given by the relation of subsumption [Ganter and Wille, 1999]. The definition of hierarchies from Ganter and Wille [Ganter and Wille, 1999] allows a concept to have more than one super-concept or sub-concept, which is closer to the real-life concept organization. This definition has been used by several researchers [Stumme and Maedche, 2001, Bozsak et al., 2002, Wille, 2002, Buitelaar et al., 2005, Cimiano and Völker, 2005, Cimiano, 2006, Bendaoud et al., 2008a, Jia et al., 2009]. We use in this thesis the definition of concept hierarchies based on [Cimiano, 2006] that fits the definition from Ganter and Wille [Ganter and Wille, 1999].

Definition 2.3 (Concept Hierarchy)**.** A *concept hierarchy* of an ontology is a structure $(H, root, \leq_H)$ consisting of (i) a set H of concept identifiers C, (ii) a designated root element representing the top element of the (iii) partial order $\leq_H$ on $H \cup \{root\}$ such that $\forall C \in H : C \leq root$.

An example of a concept hierarchy of animals is depicted in Figure 2.5. In this example, the attributes are `m1: has_two_legs`, `m2: lays_eggs`, `m3: can_fly`, `m4: has_wings`, `m5: has_fins`, `m6: has_feathers`, `m7: has_milk`, `m8: has_backbone`, and `m9: lives_in_water`. The concept hierarchy contains a root, C_0, and a set of concepts, C_1, C_2, C_3... In this concept hierarchy, the set of concepts is organized into a partial order, in which more specific concepts inherit the attributes of the more general ones. For example, concept C_5 is more specific than concept C_1 and inherits attribute `m8: has_backbone` of concept C_1; concept C_4 has two sub-concepts, concepts C_9 and C_{12}; concept C_9 has two super-concepts, concepts C_2 and C_4.

After presenting the background concepts and techniques, now, we move on to investigate the work that supports domain experts for extracting knowledge from texts, particularly in keeping the traceability and the co-evolution between the knowledge model and the source of knowledge in texts.

2.2.4 Related Work on Interactive Extracting Knowledge from Texts

Regarding the topic of supporting domain experts in extracting knowledge from texts, several researchers [Cimiano and Völker, 2005, Aussenac-Gilles et al., 2008, Szulman et al., 2010, Tissaoui et al., 2011] have been interested in keeping the traceability and the co-evolution of the knowledge model and the source of knowledge in texts.

Text2Onto, one of the pioneer frameworks for ontology learning from texts, was presented by Cimiano et al. [Cimiano and Völker, 2005, Maedche and Staab, 2001, Maedche and Staab, 2000a, Maedche and Staab, 2000b]. One of the advantages of Text2Onto is, it provides the traceability between the knowledge model and the source of knowledge in texts to help domain experts in understanding why concepts are introduced in the ontology. In order to keep the traceability, Text2Onto links objects in the ontology and terms in texts by storing a pointer for each object in the ontology to terms in texts. Another advantage of Text2Onto is, thanks to the link of objects in the ontology and terms in texts, Text2Onto is able to avoid processing the whole corpus from scratch each time it changes, only updating the corresponding parts of the ontology, i.e. the source of knowledge and the ontology co-evolve. However, Text2Onto only works with atomic concepts since in Text2Onto, concepts have no formal definitions, objects are only linked with terms.

Terminae [Aussenac-Gilles et al., 2008], a platform for ontology engineering from texts, supported domain experts in evaluating and adding knowledge. Similar to Text2Onto, Terminae

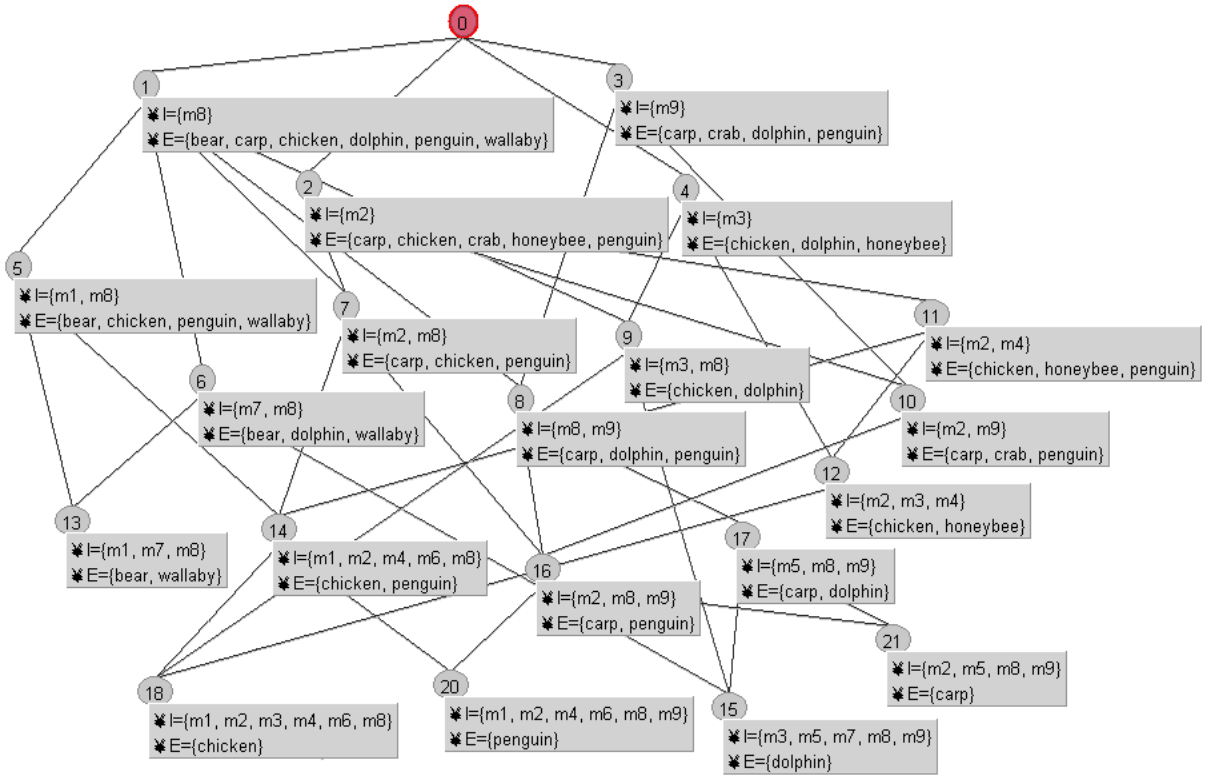

Figure 2.5: Example of a concept hierarchy of animals.

provides domain experts with the traceability between the ontology and the source of knowledge in texts. When domain experts build or maintain the knowledge model of the ontology, they could get terms for explaining why concepts are introduced in the ontology. In TERMINAE, the traceability is achieved through the links that connect terms in texts and concepts in the ontology. Several terms in texts could link to a concept in the ontology and domain experts could define or edit terms of concepts through a terminological form, then come back to modify the knowledge model. The advantage of TERMINAE is, concepts in the ontology are able to link to more than one terms in texts and have definitions from domain experts. However, the disadvantage of TERMINAE is, TERMINAE only works with atomic concepts since in TERMINAE, concepts have no formal definitions, they are defined by terms found in texts or natural language definitions that added by domain experts. Moreover, in TERMINAE, the knowledge model of the ontology and the source of knowledge do not co-evolve, domain experts have to do the task of modifying the knowledge model after they edit or add some knowledge to the source of knowledge. Then, there is a need for taking into account both atomic and defined concepts, and keeping the co-evolution of the source of knowledge and the knowledge model of the ontology to complete TERMINAE.

In the line of keeping the traceability between the knowledge model and the source of knowledge in texts with TEXT2ONTO and TERMINAE, DAFOE, a platform for building ontologies from texts, was introduced by Szulman et al. [Szulman et al., 2010]. Szulman et al. [Szulman et al., 2010] claimed that textual data cannot be mapped directly into an ontology, such as in TEXT2ONTO and TERMINAE. DAFOE uses a data model for transforming texts into ontologies which is composed of four layers: *corpora layer* for the original texts, *terminological layer* for the terms extracted from the texts, *termino-conceptual layer* for the relations of terms and concepts, and *ontology layer* for the ontology. The advantage of DAFOE is, by adding intermediate layers for formalizing the source of knowledge in texts, it allows concepts in an ontology can be independent of linguistic information in texts. Through this work, we can see that formalizing the source of knowledge in texts is also important for the process of knowledge extraction from texts. For our purpose, we can apply formalizing the source of knowledge in texts to bridge the possible gap between a knowledge model resulting from a bottom-up approach and a knowledge model from a domain expert that we have seen in section 2.2.1. Then, the source of knowledge can be modified to get a better knowledge model with respect to domain experts' requirements without touching the original texts.

Another work, EvOnto, was introduced by Tissaoui et al. [Tissaoui et al., 2011]. In EvOnto, semantic annotations are used to formalize the source of knowledge in texts. Semantic annotations are used to connect terms in texts and objects of concepts in an ontology. The advantage of EvOnto is, it manages to adapt the ontology when documents are added or withdrawn, and to keep the semantic annotations up-to-date when the ontology changes. In this system, domain experts can make changes in the ontology. However, the disadvantage of this system is, the experts cannot make changes in the source of knowledge, i.e. semantic annotations, or add background knowledge. In addition, EvOnto only works with atomic concepts since semantic annotations in this system only connect terms and objects of concepts in the ontology.

As we can see from the works above, it is important to keep the traceability and the co-evolution between the knowledge model and the source of knowledge in texts to help domain experts in the interpretation, evaluation, and refinement of the knowledge model. In addition, formalizing the source of knowledge, and taking into account both atomic and defined concepts are important as well. However, none of these works supports both the co-evolution of the knowledge model and the source of knowledge, and take into account both atomic and defined concepts. This thesis targets all these problems, using semantic annotations for formalizing the

source of knowledge and keeping the traceability between the knowledge model and the source of knowledge in texts. To take into account both atomic and defined concepts, we choose Formal Concept Analysis to be our classification method for discovering concepts with definitions of set of objects and sets of attributes and building the concept hierarchy in our knowledge extraction process. Moreover, Formal Concept Analysis is used as the key in the process to ensure the link between each knowledge units in the knowledge model and semantic annotations for keeping the traceability and the co-evolution between the knowledge model and semantic annotations.

In the following, we review the basic concepts of semantic annotations of texts, the techniques for encoding semantic annotations and linking them to texts that support us in formalizing the source of knowledge in texts and keeping the traceability between the source of knowledge and the knowledge model in our knowledge extraction process.

2.3 Semantic Annotation

2.3.1 What is Semantic Annotation?

Annotation

Annotation (or tagging) is about attaching metadata (comments, explanations, descriptions...) to some pieces of data (texts, images...) in order to provide additional information about the corresponding annotated pieces of data [4].

For our purpose, we use semantic annotation of texts.

Semantic Annotation of Texts

In 2002, in the context of semantic web, an annotation was defined by Euzenat [Euzenat, 2002] as "the content represented in a formal language and attached to the document". In 2003, Kiryakove et al. [Kiryakov et al., 2003] considered "semantic annotation is about assigning to the entities in the text links to their semantic descriptions. This sort of metadata provides both class and instance information about the entities". In 2006, in order to distinguish *semantic annotation* from the other annotations, Oren et al. [Oren et al., 2006] classified annotations into there types of annotations: *informal annotations*, which are not machine-readable; *formal annotations*, which are formally defined, i.e. machine-readable, but do not use ontological terms ; and *ontological annotations*, which are formally defined and use only ontological terms. Concerning the essential of ontologies, according to Uren et al. [Uren et al., 2006], "semantic annotation formally identifies concepts and relations between concepts of an ontology in documents, and is intended primarily for use by machines". In 2008, Lin [Lin, 2008] considered semantic annotation as "an approach to link ontologies to the original information sources". In 2009, Talantikite et al. [Talantikite et al., 2009] described semantic annotation as "an annotation assigns to an entity, which is in the text, a link to its semantic description. A semantic annotation is referent to an ontology". More recently, Bontcheva and Cunningham [Bontcheva and Cunningham, 2011] defined semantic annotation as a process of "annotating in texts all mentions of concepts from the ontology (i.e. classes, instances, properties, and relations) through metadata referring to theirs URIs in the ontology".

Figure 2.3 shows an example of semantic annotation of texts in the KIM Semantic Annotation platform[5] taken from [Kiryakov et al., 2003]. In this example, the string "XYZ" in the text is

[4] `http://www.w3.org/Amaya/User/doc/Annotations.html`
[5] http://www.ontotext.com/kim

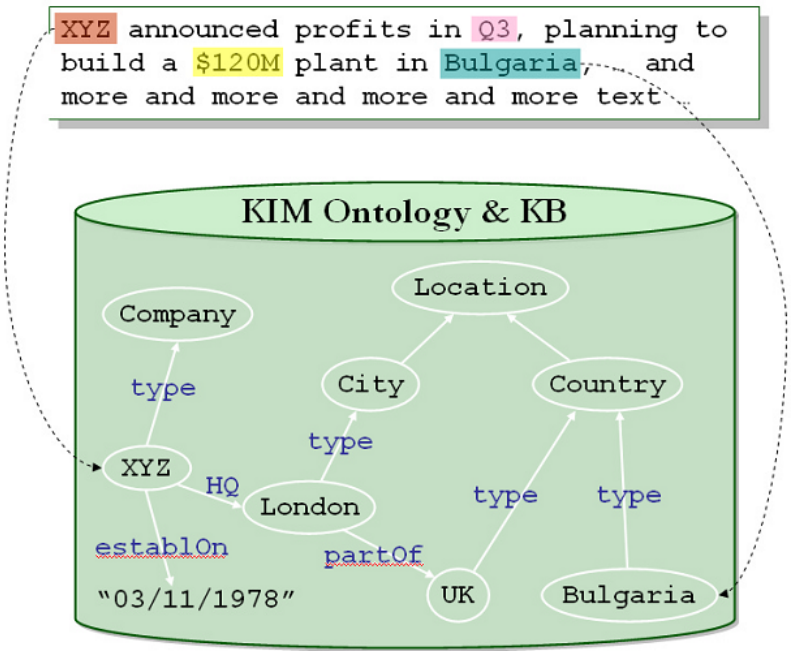

Figure 2.6: Example of semantic annotation. Figure taken from [Kiryakov et al., 2003].

linked to the object `XYZ` of the concept `Company` and the string "Bulgaria" in the text is linked to the object `Bulgria` of the concept `Country` of the ontology.

The definitions of semantic annotation from different researchers above show two things in common: semantic annotations are formally defined (machine-readable) and link ontologies to texts, which fulfill our needs of formalizing the source of knowledge in texts, and keeping the traceability between the source of knowledge and the knowledge model in the knowledge extraction process. Moreover, semantic annotation is suitable with our objective of building lightweight ontologies as Kiryakov et al. [Kiryakov et al., 2003] claimed, lightweight ontologies are more suitable for semantic annotation than heavyweight ontologies since they are sufficient for simple definitions, and allow more efficient and scalable management of the knowledge. From this perspective, in this thesis, we use the definition of a semantic annotation process as below.

Definition 2.4 (Semantic Annotation Process)**.** The *semantic annotation process* is a process of mapping terms in texts with concepts in the ontologies or concept hierarchies. The result of this mapping is formally defined metadata.

Semantic annotation can be performed manually, automatically, or semi-automatically [Reeve and Han, 2005, Bontcheva and Cunningham, 2011]. For manual semantic annotation, humans do the task annotating texts using ontologies. Examples of manual semantic annotation tools are OntoMat-Annotizer of CREAM semantic annotation framework [Handschuh and Staab, 2002], Semantic Word [Tallis, 2003], Zemanta[6], etc. Manual semantic annotation suffers from several limitations. It is considered very expensive to carry out [Cimiano et al., 2004a]. Human annotators may provide unreliable annotations due to several reasons such as the annotators are unfamiliar with the domain or the ontology schema are too complex [Bayerl et al., 2003]. Changes in ontologies can lead to semantic annotation maintenance issues [Dingli et al., 2003, Bontcheva

[6] `http://www.zemanta.com`

and Cunningham, 2011]. The annotating task can be an overwhelming task for humans because of the volume of texts [Kosala and Blockeel, 2000]. Finally, manual semantic annotation leads to a knowledge acquisition bottleneck [Maedche and Staab, 2001]. To overcome those limitations, automatic and semi-automatic semantic annotation have been proposed. Semi-automatic semantic annotation relies on humans at some point in the annotation process. For example, automatic semantic annotation systems create some annotations and then, human annotators post-edit and correct these annotations.

A (semi-)automatic semantic annotation system is composed of the following elements:

- the ontologies (or concept hierarchies) of the domain of discourse,
- the semantic annotation techniques that link terms in texts to concepts in the ontologies,
- the encoding of semantic annotations, i.e. the metadata.

We review in the following sub-sections the current semantic annotation techniques and the methods for encoding semantic annotations, and then we discuss the relation between knowledge extraction and semantic annotation.

2.3.2 Semantic Annotation Techniques

Named entity recognition (or information extraction approach) is the most common approach for (semi-)automatic semantic annotation [Bontcheva and Cunningham, 2011]. In named entity recognition, an *entity* refers to a real-world object such as "Francois Hollande, born 12 August 1954, a French politician, the current President of France, as well as Co-prince of Andorra, since 2012"[7]. A *name* (or *mention*) of an entity is a lexical expression referring to that entity in a text. For example, the entity above can be mentioned in a text by names such as "Francois Hollande" or "President Hollande". Same types of entities are grouped into *concepts* such as person, country, or organization, etc.

In named entity recognition approach, the main tasks are [Bontcheva and Cunningham, 2011]:

- Named entity recognition: This task consists of identifying the names of named entities in texts and assigning concepts to them [Grishman and Sundheim, 1996].
- Co-reference resolution: This task deals with deciding if two terms in texts refer to the same entity by using the contextual information from texts and the knowledge from ontology.
- Relation extraction: This task deals with identifying relation between entities in texts.

The techniques for identifying named entities in texts are generally classified into two groups [Sarawagi, 2008, Bontcheva and Cunningham, 2011]:

- Rule-based systems (or pattern-based systems [Reeve and Han, 2005]),
- Machine-learning systems.

Rule-based systems are based on lexicons and handcrafted rules and/or an existing list of reference entity names for identifying named entities. In these systems, named entities are identified by matching texts with the rules and/or simply doing a search through the list of

[7] `https://en.wikipedia.org/wiki/Francois_Hollande`

entity names. The rules are designed by language engineers. An example of such rules can be "[person], [office] *of* [organization]" for determining which person holds what office in what organization[8]. This rule can be used to recognize named entities of people, offices, organizations from texts such as "Vuk Draskovic, leader *of* the Serbian Renewal Movement". Some well-known rule-based systems for named entity recognition include FASTUS [Appelt et al., 1995], LaSIE [Gaizauskas et al., 1995] and LaSIE II [Humphreys et al., 1998], etc. FASTUS uses handcrafted regular expression rules to identify entity names; LaSIE and LaSIE II use lists of reference entity names and grammar rules. The advantages of rule-based systems are, they are efficient for domains where there is a certain formalism in the construction of terminology and do not require training data. The disadvantages of these systems are, they require expertise in the knowledge about language and domain, which are expensive to create [Sarawagi, 2008,Bontcheva and Cunningham, 2011]. Moreover, it is difficult to build a list of entity names that covers all the vocabularies of a domain of discourse. Therefore, they are expensive to maintain and not transferable across domains. Consequently, researchers have shifted the focus towards machine-learning based approaches since they are introduced.

In a comparison with rules-based systems, the advantage of machine-learning systems is, they automatically identify named entities. However, they require at least some annotated training data. Depend on the training data, the approaches of machine-learning systems can be further divided into three sub-groups: *supervised*, *semi-supervised* (or *weakly supervised*) and *unsupervised* methods [Nadeau and Sekine, 2007].

In supervised learning methods, the training data is an annotated training corpus, and the systems use the training data as the input of an extraction model to recognize similar objects in the new data as a way to automatically induce rule-based systems. Supervised learning methods for named entity recognition include using Support Vector Machines (SVM) ([Isozaki and Kazawa, 2002, Ekbal and Bandyopadhyay, 2010]), Hidden Markov Models ([Zhou and J.Su, 2004, Ponomareva et al., 2007]), Conditional Random Fields (CRF) ([Kazama and Torisawa, 2007, Arnold et al., 2008]), Decision Trees ([Finkel and Manning, 2009]), etc. The main disadvantages of supervised learning methods are, they require a large annotated training corpus and heavily depend on the training data, which has to be created manually. In semi-supervised learning methods, the training data are both of annotated training corpus and unannotated training corpus with a small set of examples to reduce system's dependence on the training data. The system firstly is trained with an initial set of examples to annotate the unannotated training corpus. The resulting annotations are then used to augment the initial annotated training corpus. The process repeats for several iterations to refine the semantic annotations of the documents. The disadvantage of these methods is, errors in the annotations can be propagated when the annotations are used for training the other. In unsupervised learning methods, the data is unannotated. Unsupervised learning methods use clustering techniques ([Cimiano and Völker, 2005]) to group similar objects together. The disadvantages of machine-learning systems are, it can be noisy and may require re-annotation if semantic annotation requirements change after the training data has been annotated.

Typical semantic annotation platforms include AeroDAML [Kogut and Holmes, 2001], KIM [Popov et al., 2004], MnM [Vargas-Vera et al., 2002], S-CREAM [Handschuh and Staab, 2002], etc. AeroDAML [Kogut and Holmes, 2001] is an semantic annotation tool using rule-based technique to automatically generate annotations from web pages. KIM [Popov et al., 2004] is a semantic annotation platform based on the GATE framework using rule-based technique (JAPE)

[8] `https://web.stanford.edu/class/cs124/lec/Information_Extraction_and_Named_Entity_Recognition.pdf`

for named entity recognition. MnM [Vargas-Vera et al., 2002] is a semantic annotation tool based on supervised learning technique that supports semi-automatic annotation. MnM provides users an environment to manually annotate texts, then the system learn from the set of annotated texts to recognizes new objects. Similar to MnM, S-CREAM (Semiautomatic CREAtion of Metadata) [Handschuh and Staab, 2002] automatically annotate texts by learning from a set of annotated training texts which is provided by users.

To summarize, the choice of semantic annotation technique, rules-based systems, machine-learning systems or hybrid approaches, depends on if there are language engineers available or training data, the complexity of domain and the size of the ontology [Bontcheva and Cunningham, 2011]. Current semantic annotation techniques can be noisy and may require re-annotation if the semantic annotations' requirements change. Besides, by identifying the names of named entities in texts and assigning concepts in an ontology, current semantic annotation techniques only work with atomic concepts, they do not care about defined concepts with formal definitions. For example, the current semantic annotation techniques may fail in recognizing objects of a concept `Mammal` defined by a set of attributes, `being warm blooded` and `having four legs`.

2.3.3 Encoding Semantic Annotations

Resource Description Format (RDF)[9] is a common format for encoding semantic annotation and sufficient for basic purposes of lightweight ontologies [Kiryakov et al., 2003]. There are three common approaches to encode semantic annotations in texts [Bontcheva and Cunningham, 2011]:

- Adding metadata directly to the texts' context (inline markup), with URIs pointing to the ontology.
- Adding metadata to the start/end of the texts.
- Storing metadata and texts in separate files and/or loaded within a semantic repository, metadata pointing to the texts.

In a comparison between the three approaches, the first approach make the retrieval faster when a system needs to retrieve where a concept or object is mentioned. In the first approach, metadata and texts are not independent and the system should have the authorization to modify the texts. The second approach encodes the fact that certain concepts or objects are mentioned in a particular text, but it is not able to retrieve where these concepts or objects occur in the text. The second approach requires data storage smaller than the first approach because it annotates only one of the mentions. In this approach, the system should have the authorization to modify the texts as well. In the third approach, metadata is independent of the texts. The choice of approach for encoding semantic annotations depends on if it is feasible to modify the text content.

Figure 2.7 shows an example of encoding semantic annotations. The first part of the figure shows a text, an abstract with ID 10071657 taken from PubMed[10]. The second part of the figure shows the encoding semantic annotations of the text in the first part that produced by SemRep [Rindflesch and Fiszman, 2003]. In this system, the texts and the metadata are stored separately.

Another example of semantic annotation applications is semantic wikis. Semantic wikis are an extension to be integrated in a wiki system that enable users to annotate wiki pages with

[9] `http://www.w3.org/RDF/`
[10] `http://www.ncbi.nlm.nih.gov/pubmed`

Input Text:

```
Fibromuscular dysplasia of the brachial artery--2 case reports and review of the literature.
Yet being an uncommon disease in general, the manifestation of the fibromuscular
dysplasia (FMD) in the upper extremities is exceedingly rare. Two patients with
FMD of the brachial artery are presented and the literature concerned is
reviewed. In this location the formation of microthrombi with subsequent
embolization into the periphery instead of progressive vessel stenosis seems to
be the leading pathophysiological principle. Therapeutic options are discussed.
```

Results:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SemRepAnnotation PUBLIC "-//NLM//DTD SemRep Output//EN" "http://semrep.nlm.nih.gov/DTD/SemRepXML_v1.6.dtd">
<SemRepAnnotation>
 <Document id="D00000000" text="Fibromuscular dysplasia of the brachial artery--2 case reports and review of the
literature. Yet being an uncommon disease in general, the manifestation of the fibromuscular  dysplasia (FMD) in the
upper extremities is exceedingly rare. Two patients with FMD of the brachial artery are presented and the literature
concerned is reviewed. In this location the formation of microthrombi with subsequent embolization into the periphery
instead of progressive vessel stenosis seems to be the leading pathophysiological principle. Therapeutic options are
discussed." >
 <Utterance id="D00000000.tx.1" section="tx" number="1" text="Fibromuscular dysplasia of the brachial artery--2 case
reports and review of the literature.">
  <Entity id="D00000000.E1" cui="C0016052" name="Fibromuscular Dysplasia" semtypes="dsyn" text="Fibromuscular dysplasia"
score="1000" begin="0" end="23" />
  <Entity id="D00000000.E2" cui="C1278940" name="Entire brachial artery" semtypes="bpoc" text="brachial artery"
score="1000" begin="31" end="46" />
  <Entity id="D00000000.E3" cui="C0007320" name="Case Reports Publication Type" semtypes="inpr" text="case reports"
score="884" begin="50" end="62" />
  <Entity id="D00000000.E4" cui="C0023866" name="Literature" semtypes="inpr" text="literature" score="1000" begin="81"
end="91" />
  <Predication id="D00000000.P1">
   <Subject maxDist="4" dist="1" entityID="D00000000.E2" relSemType="bpoc" />
   <Predicate type="LOCATION_OF" indicatorType="PREP" begin="24" end="26" />
   <Object maxDist="1" dist="1" entityID="D00000000.E1" relSemType="dsyn" />
  </Predication>
 </Utterance>

...
</Document>
</SemRepAnnotation>
```

Figure 2.7: Example of encoding semantic annotations.

semantic annotations. In semantic wikis, authors annotate an object represented by a wiki page with categories, datatypes, and relations, i.e. an object can be linked to other objects through relations. A category allows a user to classify pages and categories can be organized into a hierarchy. Based on the semantic annotations, wiki contents can be browsed, queried and reused in novel ways [Krötzsch et al., 2007]. In some semantic wikis such as Semantic MediaWiki (SMW) [Krötzsch et al., 2007], semantic annotations are integrated directly in the wiki texts. In some other semantic wikis such as IkeWiki [Schaffert, 2006], semantic annotations about wiki pages are stored outside the content of the pages. IkeWiki requires the load of an existing ontology.

Because the structure of semantic wikis is quite specific, in this thesis, we use semantic annotations of unstructured texts, not semantic annotations of semantic wikis. We focus on helping domain experts in building and refining the knowledge model and the source of knowledge in texts, i.e. semantic annotations. Thus, we choose the third approach for encoding semantic annotations, storing semantic annotations and texts separately since we do not modify the original texts, we only modify the source of knowledge, i.e. semantic annotations.

2.3.4 Knowledge Extraction and Semantic Annotation

Through the review of the background knowledge and semantic annotation techniques, we realize that change management is still an issue that needs to be addressed in semantic annotation. Semantic annotations are tied to one or more ontologies. Thus, if an ontology changes, then one or more semantic annotations need to be updated [Bontcheva and Cunningham, 2011]. For example, if an ontology is updated by removing a concept or an object, then the question is, which semantic annotations should be updated? As we see in section 2.3.2, the current semantic annotation techniques face the difficulty if the semantic annotation requirements change. It may require re-annotation which is expensive and time-consuming. Semantic annotation thus shares the same need of keeping the co-evolution between the knowledge model and semantic annotations with knowledge extraction.

Semantic annotation can be noisy. Even if an annotation is correct, the annotation may be useless for building the ontology with respect to experts' requirements. Euzenat [Euzenat, 2002] formalized semantic annotation as two mapping functions: *indexing* which mapping ontologies to a set of documents, and *annotation* which mapping a set of documents to ontologies, and claimed that the knowledge from the ontology makes explicit what is implicit in the annotations. Therefore, semantic annotation could be improved by the knowledge extraction process. For example, if a concept that groups the objects `chicken` and `penguin` and the attribute `has wings` in the knowledge model is not in accordance with the expert knowledge, then the set of annotations should be improved by removing or making invalid the corresponding annotations.

Another problem of semantic annotation is, taking into account both atomic and defined concepts is still challenging in semantic annotation as the current semantic annotation techniques mainly work with atomic concepts. Researchers [Buitelaar et al., 2009, Declerck and Lendvai, 2010] have tried to annotate non-atomic concepts. In 2009, Buitelaar et al. [Buitelaar et al., 2009] presented a model called LexInfo for annotating non-atomic concepts, in which additional linguistic information was associated with concept labels. They claimed that LexInfo could be used for annotating non-atomic concepts but they did not actually apply it for semantic annotation. Later, in 2010, Declerck and Lendvai [Declerck and Lendvai, 2010] proposed a proposal of merging the models from Terminae [Aussenac-Gilles et al., 2008] (see in section 2.2.4) and LexInfo [Buitelaar et al., 2009] for semantic annotation of texts. To deal with annotating non-atomic concepts, this model [Declerck and Lendvai, 2010] had three layers of description within the ontology, in which linguistic objects pointed to terms and terms pointed to concepts. This model was not actually applied for semantic annotation either. The limitation of these models [Buitelaar et al., 2009, Declerck and Lendvai, 2010] is, it's not easy to build the linguistic information for all concepts. Then there is a need for keeping the traceability between the source of knowledge in texts and the knowledge model [Cimiano and Völker, 2005, Aussenac-Gilles et al., 2008, Szulman et al., 2010, Tissaoui et al., 2011] as we have seen in section 2.2.4.

To target the problems above, we unify knowledge extraction and semantic annotation into one single process to benefit both knowledge extraction and semantic annotation. First, in order to bridge the possible gap between a knowledge model from a bottom-up approach and a knowledge model from a domain experts in the knowledge extraction process without touching the original texts, semantic annotations are used to formalize the source of knowledge in texts. In addition, semantic annotations helps the knowledge extraction process in providing the traceability between the knowledge model and the source of knowledge in texts. In a return, knowledge extraction helps in efficiently updating and improving semantic annotations by keeping the link between each knowledge units in the knowledge model and semantic annotations. Moreover, when knowledge extraction and semantic annotation are in one single process and the link be-

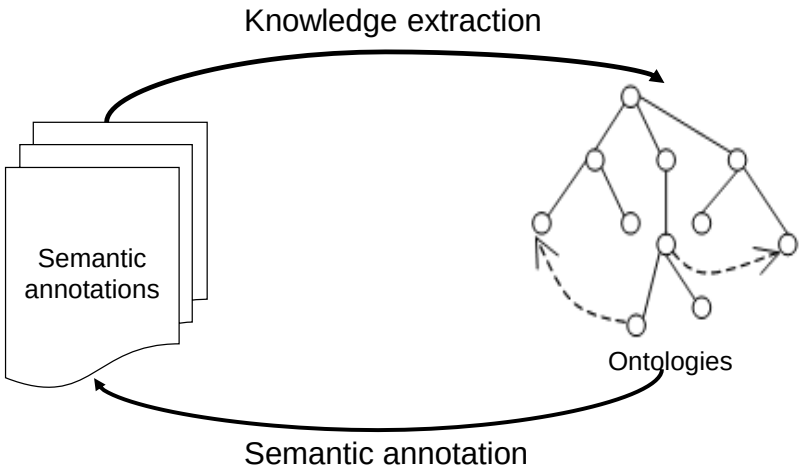

Figure 2.8: Knowledge extraction and semantic annotation loop.

tween each knowledge units in the knowledge model and semantic annotations is preserved, if we can work with both atomic and defined concept in knowledge extraction, we are able to take into account both of them in semantic annotation as well. From this perspective, knowledge extraction and semantic annotation operate in a loop as depicted in Figure 2.8. In this way, semantic annotations and ontologies are two sides of the same resource and co-evolve.

2.3.5 Choice of Data Mining Method

In this thesis, our objective is to support domain experts in extracting knowledge from texts. Formal Concept Analysis (FCA), a mathematical method based on lattice theory that can support humans to discover information and then create knowledge [Wille, 2002], perfectly fits our objective. Over the years, FCA has proved to be very efficient as a bottom-up approach for extracting knowledge from texts [Maedche and Staab, 2000a, Maedche and Staab, 2000b, Cimiano et al., 2005, Bendaoud et al., 2008a, Bendaoud et al., 2008c, Stumme, 2009, Poelmans et al., 2013]. We aim at building interactive and iterative knowledge extraction process that is able to keep the traceability and the co-evolution between the knowledge model and the source of knowledge in texts. In this context, we would like to build the ontology backbone, i.e. the concept hierarchy, and take into account both atomic and defined concepts. FCA allows us to build concepts with definitions of sets of objects and sets of attributes, and order them hierarchically at the same time, therefore, allows us to take into account both atomic and defined concepts. Another advantage of FCA is, for supporting humans in turning information into knowledge, the information should be presented in logical forms and structures, and the logical structures of concepts and concept hierarchies resulting from FCA are effective in supporting human reasoning [Wille, 2002, Napoli, 2005]. Moreover, FCA provides a good traceability in a comparison with divisive and agglomerative clusterings [Cimiano et al., 2004b, Cimiano et al., 2005]. Besides, the existing algorithms for the lattice construction can be used for real-size applications [Kuznetsov and Obiedkov, 2002, Kuznetsov, 2004, Cimiano et al., 2004b]. Thus, to achieve our objective, we choose Formal Concept Analysis to be the data mining method for our knowledge extraction process.

In the following, we present the background knowledge of Formal Concept Analysis, the data mining method that we use in this thesis.

2.4 Formal Concept Analysis

Formal Concept Analysis (FCA) is a mathematical method based on lattice theory that allows us to build concepts within a concept hierarchy for a given dataset. In the following, we first recall the basic concepts of lattice theory and then provide the theoretical foundations of Formal Concept Analysis based on [Ganter and Wille, 1999]. These definitions serve as the basic definitions that will use in the rest of the thesis.

2.4.1 Basic Concepts of Lattice Theory

Definition 2.5 (Order relation)**.** A binary relation $\leq$ on a set P is called a *order relation* (or *partial order relation*) on P if, for all, $x, y, z \in P$:

1. $x \leq x$,
2. $x \leq y$ and $x \leq y$ imply $x = y$,
3. $x \leq y$ and $y \leq z$ imply $x \leq z$.

These conditions are referred to *reflexivity*, *antisymmetry* and *transitivity* respectively.

Definition 2.6 (Ordered set)**.** An *ordered set* (or *partial ordered set* or simply *order*) is a set P equipped with an order relation $\leq$, denoted by $(P, \leq)$.

Definition 2.7 (Lattice)**.** An ordered set $(P, \leq)$ is called a *lattice* if for any pair of elements (x, y) in P the supremum $x \vee y$ and the infimum $x \wedge y$ always exist.

Supremum and infimum are also called *join* and *meet*. The supremum of all the elements in the lattice is called the *top* ($\top$), and the infimum the *bottom* ($\bot$).

Definition 2.8 (Complete lattice)**.** An ordered set $(P, \leq)$ is called a *complete lattice* if the supremum $\bigvee S$ and the infimum $\bigwedge S$ exist for any subset S of P.

Every complete lattice has a top and a bottom.

2.4.2 Formal Concept Analysis

When we use the FCA method to build the concept hierarchy of an ontology, two sets of information are needed, a set of objects and a set of attributes. The set of objects is the relevant instances and the set of attributes are used to characterize the objects which are useful for our tasks or applications. These sets are used to build a *formal context*. This formal context is then used for building a *concept lattice*, which is considered as a knowledge model.

Definition 2.9 (Formal context)**.** A *formal context* is a triple $\mathcal{K} = (G, M, I)$ where G denotes a set of objects, M a set of attributes, and I a binary relation defined on $G \times M$. $I \subseteq G \times M$ is a binary table which assigns an attribute to an object. We write gIm or $(g, m) \in I$ to mean that object g has attribute m.

Example 2.1. Table 2.1 shows a formal context of animals. In this formal context, the objects are animals (`g1`: `bear`, `g2`: `carp`, `g3`: `chicken`, `g4`: `crab`, `g5`: `dolphin`, `g6`: `honeybee`, `g7`: `penguin`, `g8`: `wallaby`) and the attributes are properties describing the animals (`m1`: `has_two_legs`, `m2`: `lays_eggs`, `m3`: `can_fly`, `m4`: `has_wings`, `m5`: `has_fins`, `m6`: `has_feathers`, `m7`: `has_milk`, `m8`: `has_backbone`, `m9`: `lives_in_water`). A cross in a position ij of the table indicates that object i has attribute j. For example, the first row in the table indicates that a `bear` `has two legs`, `has milk`, and `has backbone`.

Animal	has_two_legs (m1)	lays_eggs (m2)	can_fly (m3)	has_wings (m4)	has_fins (m5)	has_feathers (m6)	has_milk (m7)	has_backbone (m8)	lives_in_ water (m9)
bear (g1)	x						x	x	
carp (g2)		x			x			x	x
chicken (g3)	x	x	x	x		x		x	
crab (g4)		x							x
dolphin (g5)			x		x		x	x	x
honeybee (g6)		x	x	x					
penguin (g7)	x	x		x		x		x	x
wallaby (g8)	x						x	x	

Table 2.1: The binary context of animals.

Definition 2.10 (Formal concept)**.** A *formal concept* of a formal context $\mathcal{K} = (G, M, I)$ is a pair (A, B) where $A \subseteq G$ is called the *extent*, $B \subseteq M$ is called the *intent* of the concept, and A is the maximal set of objects sharing the whole set of attributes in B (and vice versa). The set of all formal concepts of the context $\mathcal{K} = (G, M, I)$ is denoted by $\mathcal{C}(G, M, I)$. A formal concept is considered to be identified by its extent and its intent.

Concepts are computed on the base of a *Galois connection* defined by two derivation operators denoted by $'$:

$$A' := \{m \in M \mid \forall g \in A, gIm\}$$
$$B' := \{g \in G \mid \forall m \in B, gIm\}$$

A concept (A, B) verifies a closure constraint so that $A' = B$ and $B' = A$.

Example 2.2. Consider the formal context given in Table 2.1, the set $A = \{\texttt{chicken}, \texttt{honeybee}, \texttt{penguin}\}$, then $A' = \{\texttt{m2 : lays\_eggs}, \texttt{m4 : has\_wings}\}$. We can see that the couple (A, B) with $B = \{\texttt{m2 : lays\_eggs}, \texttt{m4 : has\_wings}\}$ is a formal concept since $A' = B$ and $B' = \{\texttt{chicken}, \texttt{honeybee}, \texttt{penguin}\} = A$. Instead, the couple (A, B_1) with $B_1 = \{\texttt{m2 : lays\_eggs}, \texttt{m4 : has\_wings}, \texttt{m8 : has\_backbone}\}$ is not a formal concept because $A' \neq B_1$.

As we can see, formal concepts discovered by FCA (Definition 2.10) are suitable for us to build concepts with definitions in an ontology according to Definition 2.2 in section 2.2.3.

Definition 2.11 (Subsumption relation)**.** Let $C_1(A_1, B_1)$ and $C_2(A_2, B_2)$ be two formal concepts of $\mathcal{C}(G, M, I)$. Concept C_1 is a *sub-concept* of C_2 or C_2 is a *super-concept* of C_1, denoted by $C_1 \leq C_2$, if and only if $A_1 \subseteq A_2$ (*or* $B_2 \subseteq B_1$). The relation $\leq$ is called *subsumption relation* of the concepts.

Let $C_1 \leq C_2$ be a subsumption relation. We say that C_1 is subsumed by C_2.

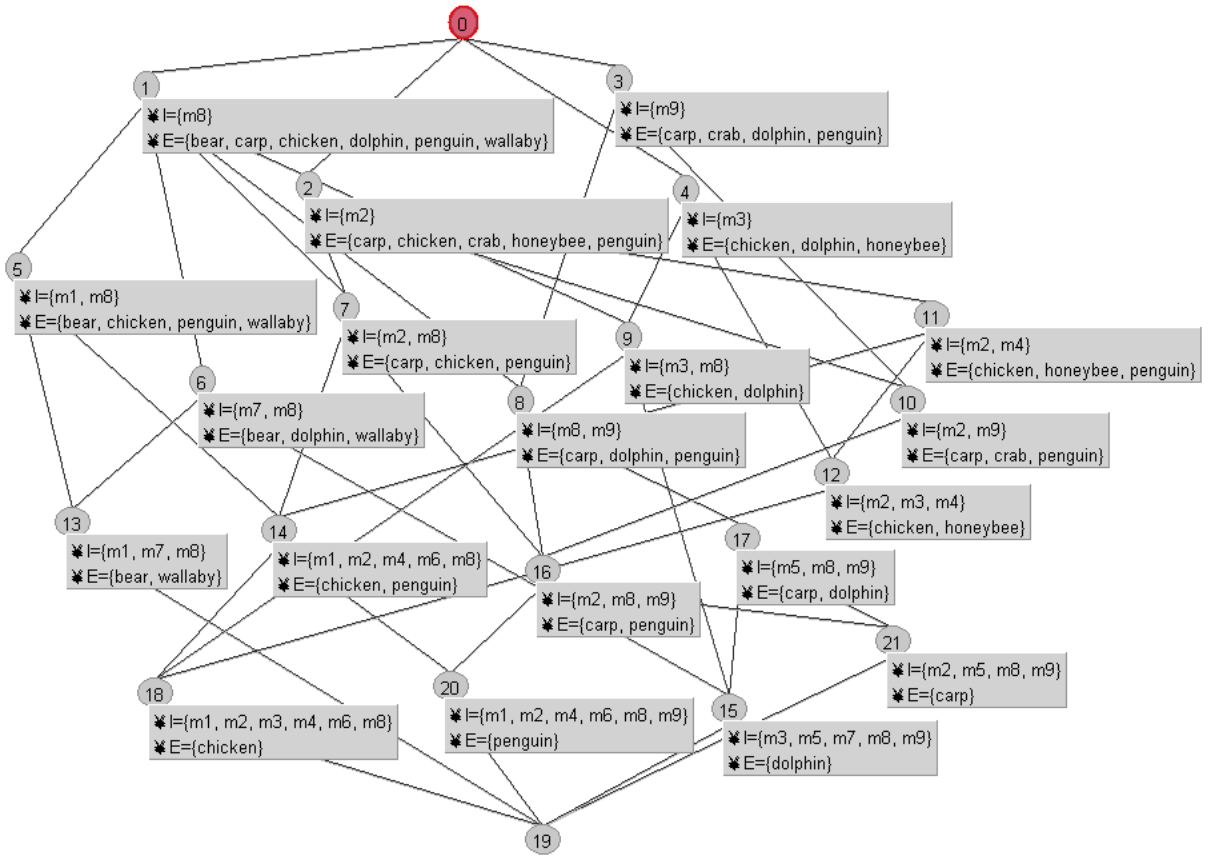

Figure 2.9: The lattice $\mathcal{L}$ built from the formal context in Table 2.1.

Definition 2.12 (Concept lattice)**.** The ordered set $\mathcal{C}(G, M, I, \leq)$ of all formal concepts of (G, M, I) is a complete lattice called the *concept lattice* of (G, M, I).

Concept lattices can be visualized as line diagrams. In a line diagram, each node represents a formal concept and an edge represents a subsumption relation between two concepts.

Example 2.3. Figure 2.9 illustrates a line diagram of the concept lattice associated to the formal context of Table 2.1. In this concept lattice, C_9 is a sub-concept of C_4 and C_4 is a super-concept of C_9. In other words, C_9 is subsumed by C_4. C_0 is the top and C_{19} is the bottom of this lattice.

The hierarchical structure of the concept lattice resulting from FCA fulfills the need for organizing concepts into a concept hierarchy of an ontology from the most general to the most specific concepts. The definitions of formal concepts in the concept lattice can be used as a base for building definitions of concepts in an ontology. The relation between concepts $C_1 \leq C_2$ in the lattice means that, an C_1 "is a" C_2. The lattice structure allows a concept to have more than one super-concept or sub-concept, which is closer to the real-life concept organization [Jia et al., 2009]. Moreover, the logical structures of formal concepts and concept hierarchies by formal concepts and concept lattices are effective in supporting human reasoning [Wille, 2002, Napoli, 2005]. The lattice is thus considered as the knowledge model where domain experts could select the concepts that fit their needs to build the final ontology. For example, the lattice shown in Figure 2.9 represents a knowledge model of an ontology about animals.

However, FCA has some disadvantages. It is sensitive to noise in the data. The presence of an irrelevant attribute in the formal context can lead to some irrelevant concepts in the concept lattice. Moreover, any formal methods has weakness when modeling a cognitive process which is not governed by clear and precise rules. As we have mentioned about the gap between a knowledge model resulting from a bottom-up approach and a knowledge model from a domain expert, domain experts may not be happy with the knowledge model, i.e. the concept lattice, produced by FCA, and wish it to be more in accordance with their requirements. For example, for concept C_{14} in Figure 2.9, the objects, `chicken` and `penguin` are grouped together because they share the common set of attributes, `m1`: `has_two_legs`, `m2`: `lays_eggs`, `m4`: `has_wings`, `m6`: `has_feathers`, and `m8`: `has_backbone`. Domain experts may not want this concept in the concept lattice because these attributes are not meaningful enough for them to characterize these objects. However, there is a unique lattice for a given dataset, if we want to change the concept lattice, we should change the data. By using semantic annotations for formalizing the source of knowledge in texts, we are able to change the lattice by changing the semantic annotations. Therefore, we are able to refine the lattice with respect to experts' requirements without changing the original texts.

2.4.3 Algorithms for Lattice Construction

Algorithms for the lattice construction can be classified into two categories: *batch algorithms* and *incremental algorithms* [Kuznetsov and Obiedkov, 2002]. Batch algorithms ([Chein, 1969, Ganter, 1984,Bordat, 1986,Kuznetsov, 1993], etc.) build the concept set and its diagram graph for the formal context from scratch. Incremental algorithms ([Norris, 1978,Godin et al., 1995,Kuznetsov and Obiedkov, 2002, Merwe et al., 2004], etc.) are used for constructing the lattice when the initial set of objects or attributes increases. Incremental algorithms, at the i^{th} step, produce the concept set or diagram graph for i first objects of the context. The $i + 1^{th}$ new object is added to the existing lattice without recomputing the whole lattice from scratch. For dynamically adding several objects at a time, [Valtchev and Missaoui, 2001] proposed an algorithm merging lattices. The existing algorithms for the lattice construction can be used for real-size applications [Kuznetsov, 2004, Cimiano et al., 2004b, Kuznetsov and Obiedkov, 2002]. Moreover, there are several implementations of these algorithms available freely online (ConExp[11], Galicia[12], ToscanaJ[13], etc.). For all the experiments in this thesis, we have developed our own system called KESAM, in which we have implemented the AddIntent incremental algorithm from [Merwe et al., 2004] for the lattice construction.

In the following, we present a survey of the related work on knowledge extraction and semantic annotation with FCA.

2.4.4 Knowledge Extraction and Semantic Annotation with FCA

On the one hand, Formal Concept Analysis (FCA) has proved to be very efficient as a bottom-up conceptualization method for knowledge extraction [Fennouh et al., 2014, Poelmans et al., 2013, Stumme, 2009, Ning et al., 2010, Jia et al., 2009, Bendaoud et al., 2008c, Bendaoud et al., 2008a, Cimiano et al., 2005, Obitko et al., 2004, Maedche et al., 2002, Wille, 2002].

Several researchers [Ning et al., 2010, Jia et al., 2009, Bendaoud et al., 2008a, Cimiano et al., 2005, Obitko et al., 2004, Maedche et al., 2002] have focused on building the concept hierarchy.

[11] `http://conexp.sourceforge.net/`
[12] `http://www.iro.umontreal.ca/~galicia/`
[13] `http://toscanaj.sourceforge.net/`

Some other researchers [Fennouh et al., 2014, Rouane-Hacene et al., 2011] built concept lattices for ontology restructuring. Maedche and several authors [Cimiano et al., 2005, Cimiano et al., 2004c, Obitko et al., 2004, Maedche et al., 2002] built a hierarchy of concepts of objects and properties extracted from texts and suggested an evaluation method by comparing the lattice to an existing thesaurus. To perform the evaluation, they defined a similarity measure. This work clearly shows that different source of information can be merged to build lattices that a hierarchy of categories (such as a thesaurus) can be introduced in the process and that the final lattice may provide definitions or quasi-definition (definition that corresponds to a superset or a subset) for these categories.

Moreover, it is possible to add background knowledge in FCA. Apposition is an operation that merges (under certain conditions) several contexts to add some background knowledge. [Stumme and Maedche, 2001] used FCA to merge two ontologies based on the context of occurrence of terms and unify terms with similar contexts. [Bendaoud et al., 2008a] made a fairly detailed presentation for the construction of ontology base on FCA. They show that the operation of apposition allows introducing some explicit knowledge to overcome the implicit knowledge contained in the texts. It is also possible to use the terms of a thesaurus to name classes, in other words, to provide a formal definition for terms from a thesaurus.

Besides, the concept lattice could be refined with respect to experts' requirements for building ontology. Researchers [Missikoff and Scholl, 1989, Carpineto and Romano, 2004, Belohlavek and Sklenar, 2005] have tried to add concepts and/or to remove irrelevant concepts to/from a concept lattice. [Missikoff and Scholl, 1989] proposed algorithms that allow users to add missing concepts to a concept lattice. [Carpineto and Romano, 2004, Belohlavek and Sklenar, 2005] have exploited additional information about the data. [Carpineto and Romano, 2004] considered domain knowledge in the form of a partial ordering relation between attributes to discard irrelevant concepts in a lattice. [Belohlavek and Sklenar, 2005] introduced the notion of attribute dependencies and used them as constraints to provide a more comprehensible structure of formal concepts. Through these works [Missikoff and Scholl, 1989, Carpineto and Romano, 2004, Belohlavek and Sklenar, 2005], we can see that, it is possible to bridge the gap between a knowledge model based on the concept lattice from FCA and a knowledge model from a domain expert.

On the other hand, FCA is also very efficient for semantic annotation [Maio et al., 2014, Shi et al., 2011, Blansché et al., 2010, Girault, 2008].

Girault [Girault, 2008] presented an unsupervised method for named-entity annotation based on concept lattices. In this approach, the formal context was built from syntactic relations between named entities in the training corpus. Formal concepts of the concept lattice built from the formal context of syntactic relation are then considered as units for named-entities annotation. The conceptual annotation was based on querying the concept lattice. For selecting the appreciate concept, the system calculate the similarity between concepts based on their intents (attributes). The result of this system shows that, supervised named entity classification is improved by using the annotation produced by their unsupervised FCA-based system.

Some researchers [Shi et al., 2011, Blansché et al., 2010] are interested in using FCA for enriching semantic wikis. In these approaches [Shi et al., 2011, Blansché et al., 2010], FCA is used for automatically computing the concept lattice that is considered as the category tree based on defined semantic properties. The category tree produced by FCA is then used for proposing changes to enrich semantic wikis,. These work shows that, the knowledge model, i.e. the concept lattice produced by FCA, can be used to improve semantic annotations.

More recently, Maio et al. [Maio et al., 2014] presents an approach for automatic semantic annotation of web resource. This approach uses a fuzzy extension of FCA and Relational Concept Analysis (RCA) [Hacene et al., 2013] to build a concept lattice from the extracted content.

The concept lattice is then translated into an ontology that used for semantic annotation. The semantic annotation of this approach got a promising performance in term of text categorization. This work confirms the fact that, knowledge extraction and semantic annotation should be in the same process, and FCA is suitable for that process.

As we can see from the works above, FCA is a very efficient method for knowledge extraction and semantic annotation. The concept lattice provides concepts with definitions of sets of objects and sets of attributes, which are useful for building and improving knowledge extraction and semantic annotation. Moreover, these works show that, it is possible to bridge the gap between a knowledge model based on the concept lattice from FCA and a knowledge model from a domain expert.

2.5 Related Work on Expert Interaction for Bridging Knowledge Gap

As we have mentioned, often there exists a gap between a knowledge model built by data mining methods and a knowledge model from domain experts in the knowledge extraction process. Concerning the problem of bridging this gap, researchers [Dzyuba et al., 2013,Dzyuba et al., 2014, Alam et al., 2015] have tried to integrate the requirements of domain experts in the information units extracted by data mining methods.

Dzyuba et al. [Dzyuba et al., 2013, Dzyuba et al., 2014] proposed a framework for interactive learning user-specific pattern interestingness. This framework goes through a loop such that, first, data mining methods are applied to extract information units (patterns) from data; then these information units are presented to users for identifying if they are interesting for users; next, the system learns from the users' feedback to get the pattern ranking with respect to the users' interests. The advantage of this framework is, it allows users to select patterns that are interesting for them among the patterns extracted by the data mining methods. However, selecting extracted patterns is not always a good option for domain experts. In some situations, a extracted pattern from data can contain both interesting and not interesting elements for domain experts. For example, for concept C_9 in the concept lattice in Figure 2.9 containing the objects `chicken` and `dolphin` and the set of attributes, `m3: can_fly` and `m8: has_backbone`, attribute `m8: has_backbone` may be interesting for domain experts to describe the objects `chicken` and `dolphin`, and attribute `m3: can_fly` may not. Then there is a need for refining or making changes in the extracted patterns with respect to experts' requirements.

Another work [Alam et al., 2015] integrated domain experts in mining definitions from annotations. Alam et al. [Alam et al., 2015] using FCA to mine implication rules from the extracted information. The implication rules are then proposed to domain experts for identifying if they can be good definitions of concepts. This approach allows domain experts to get the definitions that they want among the set of possible definitions from the extracted rules. However, experts cannot add new implication rules that do not exist in the data.

As we can see, only based on the information extracted from data, selecting among the extracted patterns [Dzyuba et al., 2013, Dzyuba et al., 2014] or the extracted rules [Alam et al., 2015] is not enough for bridging the gap between a knowledge model built by data mining methods and a knowledge model from domain experts in the knowledge extraction process. One of our objectives in this thesis is to assist domain experts in refining the extracted information to get the knowledge model as close as possible to their requirements. More precisely, we aim at allowing domain experts to create new patterns which could be more interesting than the extracted patterns by changing the source of knowledge including adding background knowledge

and new implication rules.

2.6 Summary

This chapter reviews the two key processes of knowledge-base information systems: knowledge extraction and semantic annotation. The knowledge extraction process guided by domain experts, is iterative and interactive in order to extract knowledge from data. In the knowledge extraction process, data mining methods are applied to turn data into information, and then this information is interpreted and evaluated by domain experts for becoming knowledge. Domain experts play an important role in the process of knowledge extraction: they have to decide what is considered knowledge. In the process of knowledge extraction from texts, the role of domain experts is more important and the task of interpretation and evaluation is more difficult to achieve because often there is a gap between the words used in texts and the expert knowledge. Then there is a need for assisting domain experts in interpreting, evaluating and refining the knowledge units extracted automatically from texts.

Furthermore, we provide a survey on the work that support domain experts in extracting knowledge from text, an overview of the work in this direction, Text2Onto [Cimiano and Völker, 2005], Terminae [Aussenac-Gilles et al., 2008], DaFoe [Szulman et al., 2010], EvOnto [Tissaoui et al., 2011], as well as their advantages and disadvantages. Through the survey, we realize that it is important to formalize the source of knowledge in texts, to keep the traceability and the co-evolution between the knowledge model and the source of knowledge in texts to help domain experts in interpretation, evaluation, and refinement of the knowledge model. Formalizing the source of knowledge in texts can help us in bridging the gap between the knowledge model from data mining methods and the knowledge model from domain experts in the knowledge extraction process without touching the original texts. Keeping the traceability between the knowledge model and the source of knowledge in texts make easier for domain experts in interpretation, evaluation, and refinement of the knowledge model. However, none of the work supports all these aspects, and take into account both atomic and defined concepts in the knowledge model. These approaches mainly work with atomic concepts since concepts are only connected to by terms and/or natural language definitions, i.e. concepts have no formal definitions. Therefore, we need a means for formalizing the source of knowledge in texts, keeping the traceability and the co-evolution between the knowledge model and the source of knowledge in texts. Moreover, in order to take into account and both atomic and defined concepts in the knowledge model and semantic annotations, we need a method for building concepts with formal definitions.

In order to reach our objective, we use semantic annotations to formalize the source of knowledge in texts and keep the traceability between the knowledge model and the source of knowledge. Knowledge extraction can get benefits from semantic annotation to get the traceability since semantic annotations are formally defined and link ontologies to texts. Through the review of semantic annotation, we realize that semantic annotation shares the same need of keeping the co-evolution between the knowledge model and semantic annotations with knowledge extraction. It is expensive and time-consuming if knowledge extraction and semantic annotation are treated as two separated processes when the knowledge model frequently changes, then semantic annotations frequently need to be updated, and vice versa. Moreover, semantic annotations can be noisy or useless for building ontology, they could be improved by the knowledge extraction process. Besides, taking into account both atomic and defined concepts in semantic annotation should be considered as the current semantic annotation techniques mainly work with atomic

concepts. To target all these problems, we unify knowledge extraction and semantic annotation into one single process and keep the link between each knowledge unit in the knowledge model and semantic annotations. In this way, knowledge extraction and semantic annotation can benefit from each other, we are able to bridge the gap between the knowledge model from data mining methods and the knowledge model from domain experts in the knowledge extraction process without touching the original texts; the traceability and the co-evolution between the knowledge model and semantic annotations can be enable.

To take into account both atomic and defined concepts in the knowledge extraction process, we choose Formal Concept Analysis (FCA) to be our data mining method for building the concept hierarchy and ensuring the link between each knowledge unit in the knowledge model and semantic annotations since FCA discovers concepts with definitions of sets of objects and sets of attributes, and order them hierarchically at the same time. FCA provides a good traceability in a comparison with divisive and agglomerative clusterings [Cimiano et al., 2004b,Cimiano et al., 2005], and the existing algorithms for the lattice construction can be used for real-size applications [Kuznetsov, 2004, Cimiano et al., 2004b, Kuznetsov and Obiedkov, 2002]. The related work on knowledge extraction and semantic annotation with FCA has proved that FCA is an efficient method in knowledge extraction and semantic annotation. Moreover, FCA is suitable to improve knowledge extraction and semantic annotation. However, as bottom-up approaches, often there exists a gap between the knowledge model based on a concept lattice and the knowledge model from domain experts. Domain experts may not be satisfied with the concept lattice, produced by FCA, and wish it to be more in accordance with their requirements. An approach should be proposed to bridge this gap by helping domain experts in refining the lattice. However, there is a unique concept lattice for a given dataset, if we want to make some changes in the concept lattice, we should change the data. By using semantic annotations for formalizing the source of knowledge in texts, we are able to change the lattice by changing semantic annotations. Therefore, we are able to refine the lattice with respect to experts' requirements without changing the original texts.

All these elements help us to build the iterative and interactive process of knowledge extraction from texts, which is the objective of this thesis.

Chapter 3

KESAM: A Tool for Knowledge Extraction and Semantic Annotation Management

Contents

3.1 Introduction

In this chapter, we present our methodology for interactive and iterative extracting knowledge from texts - the KESAM system: A tool for Knowledge Extraction and Semantic Annotation Management. KESAM is based on Formal Concept Analysis for extracting knowledge from textual resources that supports domain experts in evaluating and refining the knowledge model based on the concept lattice. The KESAM system also aims at improving the semantic annotation

process: annotations in texts can be modified or enhanced to make the resulting knowledge model based on the concept lattice as close as possible to the requirements of domain experts.

In the KESAM system, knowledge extraction and semantic annotation are unified into one single process to benefit both knowledge extraction and semantic annotation. Semantic annotations are used for formalizing the source of knowledge in texts and keeping the traceability between the knowledge model and the source of knowledge. This provides with domain experts a way to trace back from the knowledge model to the source of knowledge in texts to understand why concepts are built. The knowledge model is, in return, used for improving semantic annotations. The KESAM process has been designed to permanently preserve the link between the resources (texts and semantic annotations) and the knowledge model. The core of the process is Formal Concept Analysis that builds the knowledge model based on the concept lattice, and ensures the link between the knowledge model and annotations. In order to get the resulting lattice as close as possible to domain experts' requirements, we introduce an iterative process that enables expert interaction on the formal context, the lattice, and annotations. Experts are invited to evaluate and refine the concept lattice; they can make changes in the formal context, the lattice or annotations until they reach an agreement between the knowledge model based on the concept lattice and their own knowledge or application's need. Thanks to the link between the knowledge model and semantic annotations, the knowledge model and semantic annotations can co-evolve in order to improve their quality with respect to domain experts' requirements. Moreover, by using FCA to build concepts with definitions of sets of objects and sets of attributes, the KESAM system is able to work with both atomic and defined concepts, i.e. concepts that are defined by a set of attributes.

This chapter is organized as follows. Firstly, we present an overall of the KESAM methodology. In this part, we define the features to distinguish KESAM from the other work and then, we present the KESAM process. Next, we describe in detail all the steps of the KESAM process, how to use semantic annotations for formalizing the source of knowledge in texts, building concept the lattice, and tracing back from the lattice to semantic annotations. We then show how to assist domain experts in interacting for the evaluation and refinement of the knowledge model based on the lattice and keeping the consistency between the lattice and its semantic annotations. After that, we present the implementation of the KESAM system and a case study on medical domain. Finally, we include a conclusion of our work.

3.2 The KESAM Methodology

In this section, we first define the main features that verify our KESAM methodology and distinguish KESAM from the other work, and then position the KESAM methodology in the relation with the other methodologies in the direction of supporting domain experts in extracting knowledge from texts. Finally, we present the KESAM process that is designed for the KESAM system.

3.2.1 The KESAM's Features

The overall objective of the KESAM system is to support domain experts in extracting knowledge from textual resources using Formal Concept Analysis as discussed in section 2.2.2. For this purpose, KESAM focuses on the following features:

1. Providing the traceability between the knowledge model and the source of knowledge in the text: From each concept in the knowledge model, experts can trace back to the source

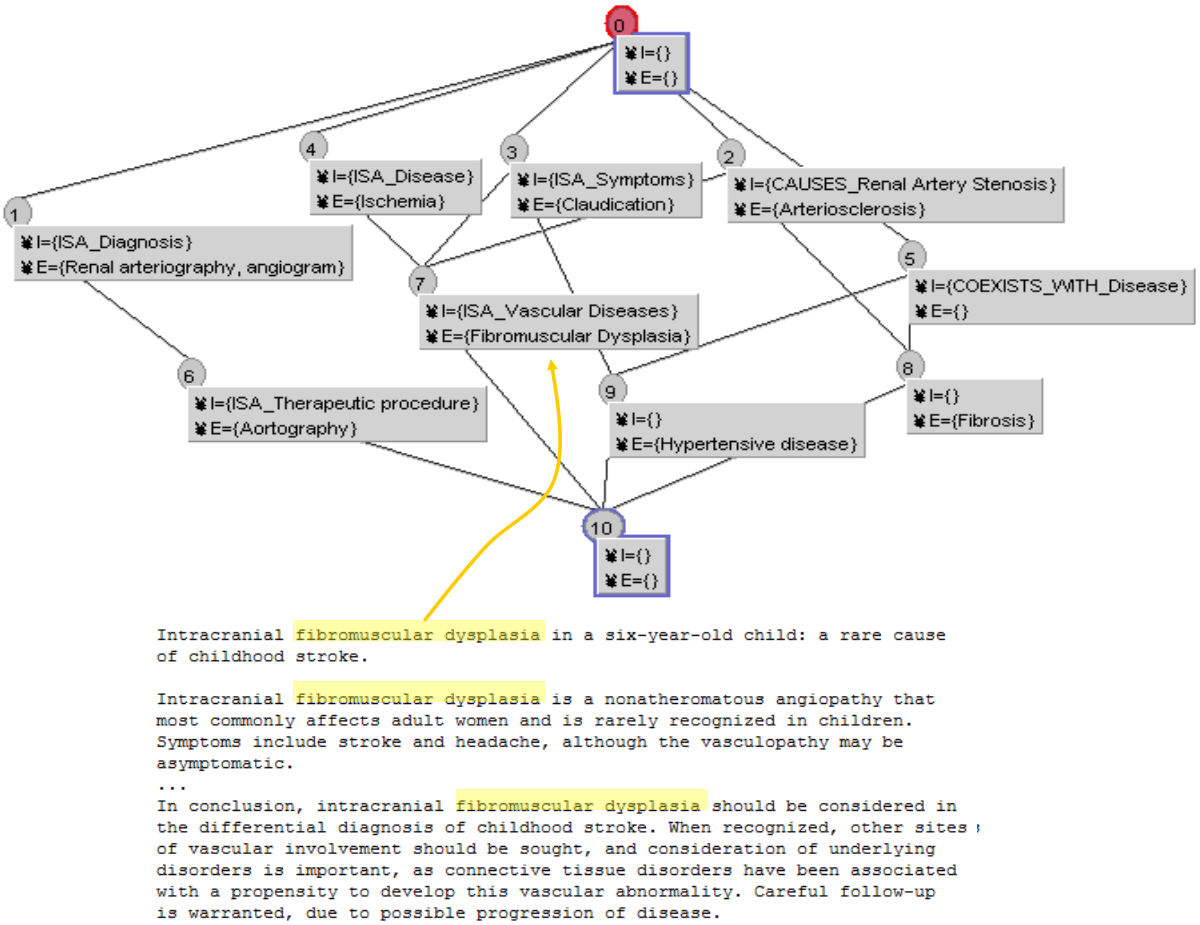

Figure 3.1: An example of the traceability between the knowledge model and source of knowledge in the text

of knowledge in texts to understand why this concept is introduced. This feature is used for helping domain experts in the evaluation and refinement of the knowledge model based on the concept lattice. Figure 3.1 illustrates an example of the traceability between the knowledge model based on the concept lattice and the source of knowledge in texts. The first part of the figure shows a knowledge model based on the concept lattice and the second part of the figure shows an abstract extracted from PubMed[14] (PMID 10961798), one of the documents that was used as a resource for building the knowledge model in the first part of the figure. With the traceability between the knowledge model and the source of knowledge in texts, experts can know where they get the information of object `Fibromuscular Dysplasia` of concept C_7 in the document, i.e. the texts are colored yellow in the second part of Figure 3.1 .

2. Enabling expert interaction: Domain experts control the knowledge extraction process and they should be able to make changes in the knowledge model, i.e. the concept lattice produced by FCA, to get the knowledge model as close as possible to their own knowledge or application's need. For example, if experts are not pleased with some concepts in the

[14] `http://www.ncbi.nlm.nih.gov/pubmed`

lattice, they can ask for removing these concepts from the lattice. Once experts ask for some changes in the lattice, the system should be able to keep the semantic annotations up to date with the lattice. Moreover, as described in section 2.2.2, experts should be able to add background knowledge or edit the source of knowledge to bridge the possible gap between and the symbols extracted from texts and the knowledge from experts. For example, in the document in the second part of Figure 3.1, there is no mention indicate that `fibromuscular dysplasia` is a disease, this information is assumed as background knowledge. Thus, the system should be able to allow experts to add this background knowledge for improving the knowledge model to be more comprehensive. Once experts add some background knowledge or edit the source of knowledge, the system should be able to keep the lattice and its annotations up to date with these changes.

3. Formalizing the source of knowledge.

4. Keeping the co-evolution between the knowledge model and the source of knowledge: Any changes in the source of knowledge reflexes in the knowledge model, and vice versa. In other words, the knowledge model is efficiently updated when the source of knowledge changes, and conversely.

5. Unifying knowledge extraction and semantic annotation into one single process: This feature aims at improving the quality of semantic annotations in accordance with the construction of the knowledge model. Semantic annotations are used as the input for building the knowledge model. The knowledge model is, in return, used to update the semantic annotations of texts. The set of annotations, thus could be improved after domain experts provide the valuable knowledge to the knowledge model.

6. Taking into account both atomic and defined concepts: The system is able to work with concepts that are defined by a set of attributes as introduced in section 2.2.3.

Table 3.1 depicts the position of the KESAM in the relation with the other methodologies in the direction of supporting domain experts in extracting knowledge from texts. The related works on interactive knowledge extraction from texts include Text2Onto [Cimiano and Völker, 2005], Terminae [Aussenac-Gilles et al., 2008] Dafoe [Szulman et al., 2010] and EvOnto [Tissaoui et al., 2011]. These related works are described in detail in section 2.2.4.

3.2.2 The KESAM Process

The KESAM process for iterative and interactive knowledge extraction from texts is based on the knowledge extraction process presented in section 2.2.1 and the data mining method Formal Concept Analysis presented in section 2.4. The KESAM process aims at building the concept hierarchy of an ontology based on the concept lattice as described in 2.2.3. In KESAM, knowledge extraction and semantic annotation are unified into one single process to benefit both knowledge extraction and semantic annotation as discussed in section 2.6. The first step of the KESAM process is about to use semantic annotations for formalizing the source of knowledge in texts. These semantic annotations are used to build the knowledge model and to keep the traceability between the knowledge model and the source of knowledge in texts. Then, Formal Concept Analysis, the data mining method that we choose for building the concept hierarchy of the ontology, is used to build the knowledge model, i.e. the concept lattice, from the formal context derived by the semantic annotations. The KESAM process follows a multi-loop composed of two main paths: one builds a knowledge model based on the concept lattice from semantic

Methodology	Providing the traceability between the knowledge model and the source	Enabling expert interaction on the knowledge model	Adding background knowledge or editing the source of knowledge	Formalizing the source of knowledge	Keeping the co-evolution between the knowledge model and the source	Unifying knowledge extraction and semantic annotation	Taking into account both atomic and defined concepts
TEXT2ONTO	x	x			x		
TERMINAE	x	x	x				
DAFOE	x			x	x		
EVOONTO	x	x		x	x	x	
KESAM	x	x	x	x	x	x	x

Table 3.1: KESAM and the other methodologies with their features.

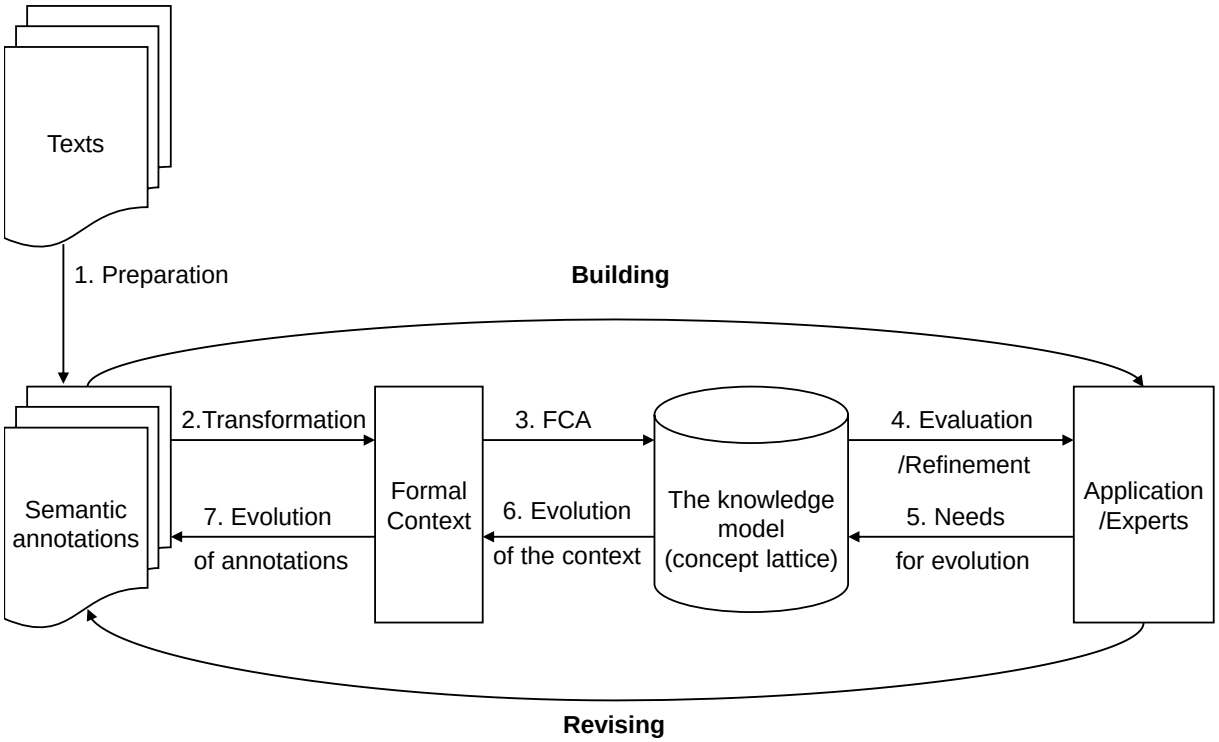

Figure 3.2: The KESAM Process

annotations, the other revises semantic annotations according to experts' requirements. Experts interact in the knowledge model based on the concept lattice, but changes are performed on the semantic annotations without touching the original texts.

The KESAM process is presented in Figure 3.2. More precisely, the steps of the KESAM process are:

1. **Preparation:** In this step, texts are first semantically annotated to identify objects and their attributes of the domain. These semantic annotations provide the set of objects and the set of attributes that are used as the input for building the knowledge model based on the concept lattice, providing the traceability between the knowledge model and the source of knowledge in texts, as well as keeping the link between them.

2. **Transformation:** In this step, the building path starts the loop with transforming the semantic annotations from step 1 into a formal context that is the input for FCA.

3. **FCA:** This step uses FCA to discover concepts with sets of objects and sets of attributes, and to build a concept lattice from the formal context from step 2. The concept lattice resulting from FCA is then considered as the knowledge model. By using FCA to build concepts with definitions of sets of objects and sets of attributes, the KESAM system is able to take into account both atomic and defined concepts, i.e. concepts that are defined by a set of attributes.

4. **Evaluation/Refinement:** In this step, domain experts are asked to evaluate and refine the knowledge model based on the concept lattice from step 3. To facilitate the evaluation, the system provides experts with the traceability between the lattice and the source of

knowledge in texts. From each concept in the lattice, experts can trace back to the texts to see why this concepts is introduced. If experts are not pleased with something in the lattice, they can refine the lattice by asking for some changes in the formal context or the lattice. Moreover, experts can make changes in annotations or add new annotations as background knowledge. The system provides experts with a pre-defined set of changes for refining the formal context, the lattice and annotations. For example, experts can ask for *removing object g from the formal context, removing concept C from the lattice, adding a new annotation of object* g_1 *and property* m_1... [Tang and Toussaint, 2013].

5. **Needs for evolution:** When experts ask for a change in the formal context, the lattice or annotations, the revising path starts. At this step, to execute a change in lattice, the system exploits the formal properties of the lattice to determine all the possible *strategies* with their *consequences*, and suggests them to experts. A change strategy can lead to some other changes in the lattice. For example, removing a concept in the lattice or removing an annotation can lead to modify, delete some other concepts or create some new concepts. These induced changes are called *consequence* of a change strategy. The consequence of a change strategy on the lattice is reported to the experts before it is actually applied. It is important for the experts to know the consequence of a change strategy because it can lead to delete some important concepts for them. Depending on the expert knowledge or application's need, experts will decide which change strategy should be applied.

6. **Evolution of the context:** In this step, when experts have chosen one of the strategies, the chosen strategy is then propagated to the formal context.

7. **Evolution of annotations:** In this step, semantic annotations are updated according to the changes in the lattice to ensure that, at the next iteration, the lattice gives a new knowledge model, closer to the experts' requirements.

The KESAM process is iterative, experts can make changes in the formal context, the lattice or annotations until they reach an agreement between the model and their own knowledge or application's need. Thanks to the link between the knowledge model based on the lattice and semantic annotations, the corpus does not have to get re-processed, only the corresponding annotations are updated. In this way, the knowledge model based on the lattice and semantic annotations can co-evolve in order to improve their quality with respect to the experts' requirements.

In the following sections, we will describe in detail all the steps of the KESAM process.

3.3 Semantic Annotations for Formalizing the Source of Knowledge, Building Lattices and Providing the Traceability

In this section, we first present a way of using semantic annotations for formalizing the source of knowledge in texts (step 1 in Figure 3.2). Next, we show how to build the concept lattice from semantic annotations (step 2 and step 3 in Figure 3.2). Then, we show how the traceability between the lattice and the source of knowledge in texts can be enabled in the KESAM system.

3.3.1 Semantic Annotations for Formalizing the Source of Knowledge

In order to use semantic annotations for formalizing the source of knowledge in texts for building the knowledge model based on the concept lattice by FCA, we need two sets of information, a

```
Intracranial fibromuscular dysplasia in a six-year-old child: a rare cause
of childhood stroke.

Intracranial fibromuscular dysplasia is a nonatheromatous angiopathy that
most commonly affects adult women and is rarely recognized in children.
Symptoms include stroke and headache, although the vasculopathy may be
asymptomatic.
...
In conclusion, intracranial fibromuscular dysplasia should be considered in
the differential diagnosis of childhood stroke. When recognized, other sites
of vascular involvement should be sought, and consideration of underlying
disorders is important, as connective tissue disorders have been associated
with a propensity to develop this vascular abnormality. Careful follow-up
is warranted, due to possible progression of disease.
```

Figure 3.3: An abstract extracted from PubMed (pmid 10961798).

set of objects and a set of attributes (see in section 2.4). At the preparation step of the KESAM process (step 1 in Figure 3.2), texts are semantic annotated to identify objects and their attributes of the domain. At this step, the semantic annotation process can be done manually or by any automatic tool. Whatever is the quality of the annotation process, it is the initial set of input data. The aim of the KESAM system is to improve the quality of the annotations in accordance with the construction of the knowledge model based on the concept lattice. In KESAM, domain experts are invited to evaluate and refine the lattice, and the source of knowledge, i.e. semantic annotations, is improved in accordance with the construction of the lattice. For this purpose, we do not modify the original texts, we only modify the semantic annotations.

Our experiment focused on texts in the field of medicine. Texts were extracted from PubMed[15]. We used SemRep [Rindflesch and Fiszman, 2003] to annotate the texts. SemRep identifies entities that can be found in the Unified Medical Language System (UMLS) Metathesaurus. SemRep also provides some additional information about entities such as the preferential UMLS terms and their position in the texts. The relations between concepts are also extracted. Figure 3.3 shows an example of an abstract extracted from PubMed. Figure 3.4 shows annotations resulting from SemRep for the text in Figure 3.3. In this example, `Fibromuscular Dysplasia` is extracted as an entity associated with the concept `C0016052` in UMLS and `Fibromuscular Dysplasia` has relation `AFFECTS women` (the last line of the figure). In the context of FCA, to deal with relational attributes, we scale them and treat them as normal attributes in the lines of [Rouane-Hacene et al., 2007]. In the KESAM system, to build the binary context, given an annotation ($object_1$, $relation$, $object_2$), we consider $object_1$ as an object and the name of the relation is concatenated with $object_2$ to become a binary attribute of $object_1$. In this example, `Fibromuscular Dysplasia` is considered as an object and `AFFECTS_women` is considered as one of its attributes. SemRep annotation process can be noisy as no automatic tools is perfect. However, any annotation process can be used to annotate texts, including manual annotation. This is one of our goals to improve the annotations in parallel with the construction of the knowledge model based on the concept lattice.

In order to provide the traceability between the knowledge model based on the concept lattice and the source of knowledge in texts, in the KESAM system, besides the information about objects and their attributes, the additional information about their positions and original

[15] `http://www.ncbi.nlm.nih.gov/pubmed`

```
SE|00000000||tx|1|text|Intracranial fibromuscular dysplasia is a
nonatheromatous angiopathy that most commonly affects adult women
and is rarely recognized in children.
```

- ...
- SE|00000000||tx|1|entity|C0016052|Fibromuscular Dysplasia|dsyn|||fibromuscular dysplasia||||901|14|36
- SE|00000000||tx|1|entity|C0042373|Vascular Diseases|dsyn|||angiopathy||||861|59|68
- SE|00000000||tx|1|entity|C0043210|Woman|popg|||women||||888|103|107

  ...
- SE|00000000||tx|1|relation|0|0|C0016052|Fibromuscular Dysplasia|dsyn|dsyn|||fibromuscular dysplasia||||901|14|36|INFER|AFFECTS(SPEC)||14|68|0|0|C0043210|Woman|popg,humn|humn|||women||||888|103|107

Figure 3.4: Entities and relationships extracted by SemRep for the given text in Figure 3.3.

strings in texts is also kept in the annotations.

Moreover, in the KESAM system, we keep the status of annotations. The status of an annotation in KESAM can be *valid* or *invalid*. If an annotation is considered as wrong or not meaningful, then its status will be *invalid*. Otherwise, the status of an annotation is *valid*. All annotations with status *valid* or *invalid* are kept in the database, but only annotations with status *valid* are used to build the knowledge model. Annotations with status *invalid* are still kept in the database for using in case that experts recognize that they make a wrong decision before, they can update the status of those annotations to become *valid*.

Formally, an annotation in the KESAM system is defined as follows.

Definition 3.1 (Annotation)**.** An *annotation* in the KESAM system is a tuple $< g, m, P, T, s >$, where g is an object that has the attribute m, P its positions, and T its terms in a corpus, s its status in KESAM. The value of s can be *valid* or *invalid*.

These annotations are stored in Resource Description Framework (RDF) format, and independent with the original texts as described in section 2.3.3. Figure 3.5 shows an example of the RDF representation format that used for representing semantic annotations in KESAM. In this example, the semantic annotation contains an object `fibromuscular_dysplasia` that has an attribute `occurs_in_elderly_man`. This annotation also contains the information of the positions of the object and the attribute in the corpus, and its status is *valid*.

```
<rdf:Description rdf:about="http://loria.fr/test/an_333">
        <Annotation:Object>fibromuscular_dysplasia</Annotation:Object>
        <Annotation:Attribute>occurs_in_elderly_man</Annotation:Attribute>
        <Annotation:Position rdf:parseType="Resource">
                <Annotation:subEnd>1049</Annotation:subEnd>
                <Annotation:subStart>1020</Annotation:subStart>
                <Annotation:objectStart>985</Annotation:objectStart>
                <Annotation:objectEnd>996</Annotation:objectEnd>
                <Annotation:relationEnd>1001</Annotation:relationEnd>
                <Annotation:relationStart>997</Annotation:relationStart>
                <Annotation:text>
                We present the first case, to our knowledge, of an elderly man with
                ....
                </Annotation:text>
                <Annotation:file>...</Annotation:file>
        </Annotation:Position>
        <Annotation:Terms>
                fibromuscular dysplasia
        </Annotation:Terms>
        <Annotation:Status>valid</Annotation:Status>
</rdf:Description>
```

Figure 3.5: An example of RDF representation in KESAM.

3.3.2 Building Lattices and Providing the Traceability with Semantic Annotations

After using semantic annotations to formalize the source of knowledge in texts, we apply FCA to build the knowledge model based on the concept lattice from these semantic annotations (step 2 and step 3 - Figure 3.2).

Annotations with status *valid* provide the set of objects G and the set of attributes M that are transformed into a formal context $\mathcal{K} = (G, M, I)$; I is the relation where $I(g, m)$ is a statement that g has the attribute m (step 2 - Figure 3.2). From this formal context, we obtain the concept lattice resulting from FCA (step 3 - Figure 3.2). Figure 3.6 illustrates an example of building the concept lattice from semantic annotations of texts.

The construction of the concept lattice from the formal context is described in section 2.4. The concept lattice resulting from FCA is then considered as the knowledge model and proposed to domain experts for the evaluation and refinement. Table 3.2 illustrates a formal context $\mathcal{K}$ describing a set of objects about diseases and their attributes. The corresponding lattice $\mathcal{L}$ is given in Figure 3.7.

The annotation process in the KESAM system annotates a concept in the concept lattice with any occurrence of objects contained in the concept extent and the attributes in the concept intent. In the concept lattice, a concept is defined by a set of objects (extent) and attributes (intent). When a domain expert want to trace back from a concept C in the concept lattice to the source of knowledge in texts. The system then queries all the annotations with status *valid* w.r.t. the objects the concept extent and the attributes in the concept intent of concept C. With the information of positions in an annotation (Definition 3.1), the system is able to provide the source of knowledge in the texts for each pair of object and attribute and thus, for a concept. An example of tracing back from a concept in the concept lattice to the source of knowledge in texts is shown in Figure 3.8. In this example, an expert trace back from concept C_{35} with extent `Hypertensive disease` and intent {`dysn, PROCESS_OF_Woman, COEXISTS_With_Old age`}

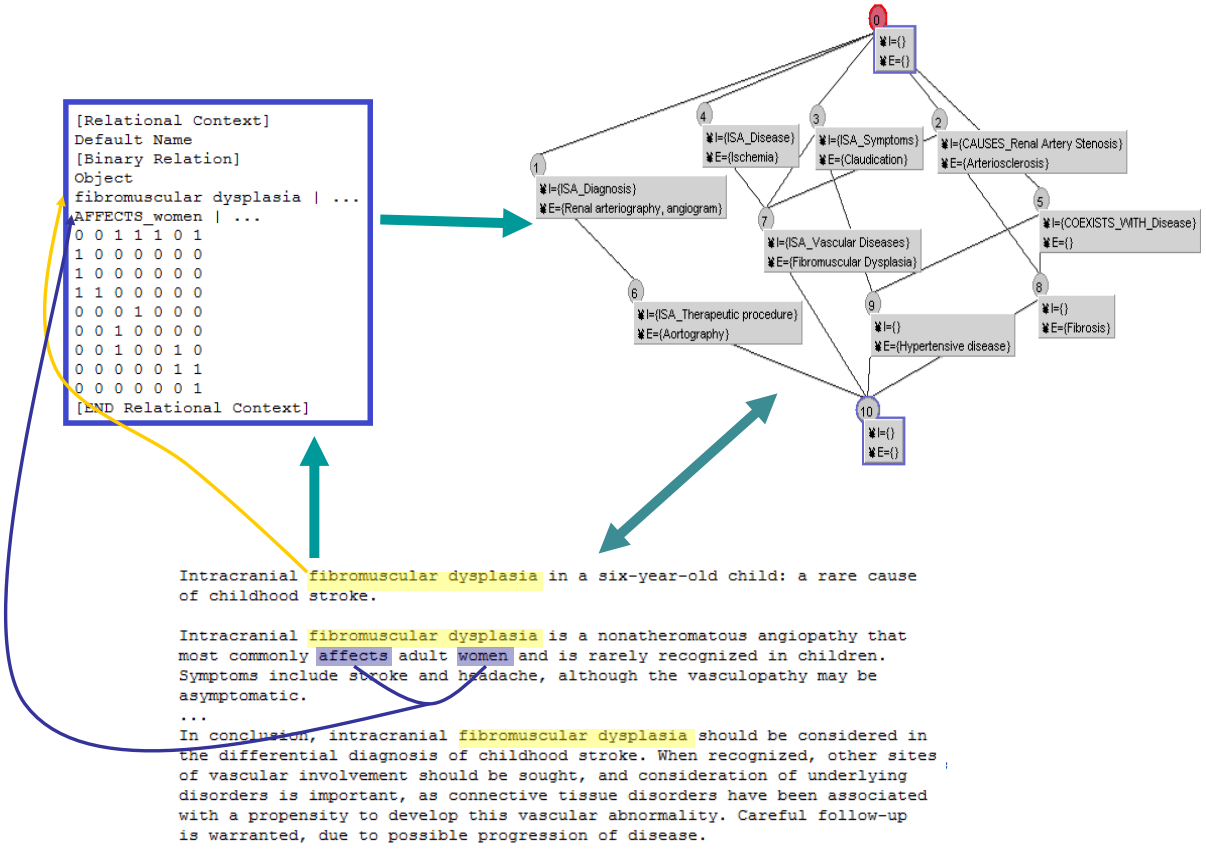

Figure 3.6: An example of building a lattice from semantic annotations of texts.

in the concept lattice to the source of knowledge in texts. The KESAM system provides her with the source of knowledge in texts of the object that is colored yellow and the source of knowledge in texts of the attributes that is colored in green.

When new documents are introduced in the corpus, the system searches for object occurrences w.r.t to the information of terms in the set of *valid* annotations (Definition 3.1). The system then annotates these object occurrences with the concepts in the lattice that these objects in its extent. To detect new objects, the system searches for attribute occurrences. If the whole set of attributes in a concept intent is found in texts and refers to the same linguistic entity, this linguistic entity may be an object corresponding to this concept. KESAM considers new objects are linguistic entities that a whole set of attributes in the intent of a concept in the lattice refers to. In this way, the system can work for annotating objects corresponding to defined concepts. For instance, the occurrences of the set of attributes, *disease or syndrome* (`dysn`) and *concerning women* (`PROCESS_OF_Woman`) (C_{23} in Figure 3.11), are found in texts, and they refer to the same linguistic entity *gallstones*. *gallstones* is thus considered as a new object corresponding to concept C_{23}.

Object	OCCURS_IN_woman	ISA_rare_disease	COEXISTS_WITH_old_age	CAUSES_ischemia
fibromuscular_dysplasia	x	x		x
breast_disease	x			
hypertensive_disease	x		x	
bone_disease			x	x

Table 3.2: Binary context $\mathcal{K}$.

3.4 Expert Interaction for Evaluation and Refinement

In KESAM, the knowledge model based on the concept lattice is built iteratively and domain experts are invited to evaluate and refine it (step 4 in Figure 3.2). However, FCA is sensitive to noise. Noise in annotations can lead to noise in the lattice. Moreover, as mentioned in section 2.2.2 and 2.4, several reasons may motivate experts ask for changes in the knowledge model based on the concept lattice. Improving the model depends on experts' understanding of the domain or applications' need. For example, in Figure 3.7, concept C_6 corresponds to a concept of diseases that concern old women. In this concept, the intent contains two attributes, concerning women (`PROCESS_OF_Woman`), and concerning old people (`COEXISTS_WITH_Old age`), the extent contains one object `hypertensive disease`. This information is extracted from texts, but it is wrong from experts' point of view. Experts may consider this as an over-specific concept as `hypertensive diseases` do not only concern women. If experts consider that attribute `concerning women` should not be an attribute of object `hypertensive disease`, then they may want to ask for removing this attribute from object `hypertensive disease`.

To assist domain experts in the evaluation and refinement of the knowledge model based on the concept lattice and its semantic annotations, KESAM needs to be addressed the two following questions:

1. How domain experts can specify their wishes of changes to refine the formal context, the concept lattice and its semantic annotations?

2. How these changes can be applied to the formal context, the concept lattice and its semantic annotations?

In this section, we present an approach for assisting domain experts in specifying their wishes of changes for refining the formal context, the lattice and semantic annotations, and then applying these changes to the formal context, the concept lattice and semantic annotations.

To assist domain experts in specifying their wishes for refining the formal context, the concept lattice and semantic annotations, we provide them with a set of changes (step 5 in Figure 3.2). In the following, we provide the formulations of the changes for refining the formal context, the concept lattice, and semantic annotations in the KESAM system. Then, we show how these

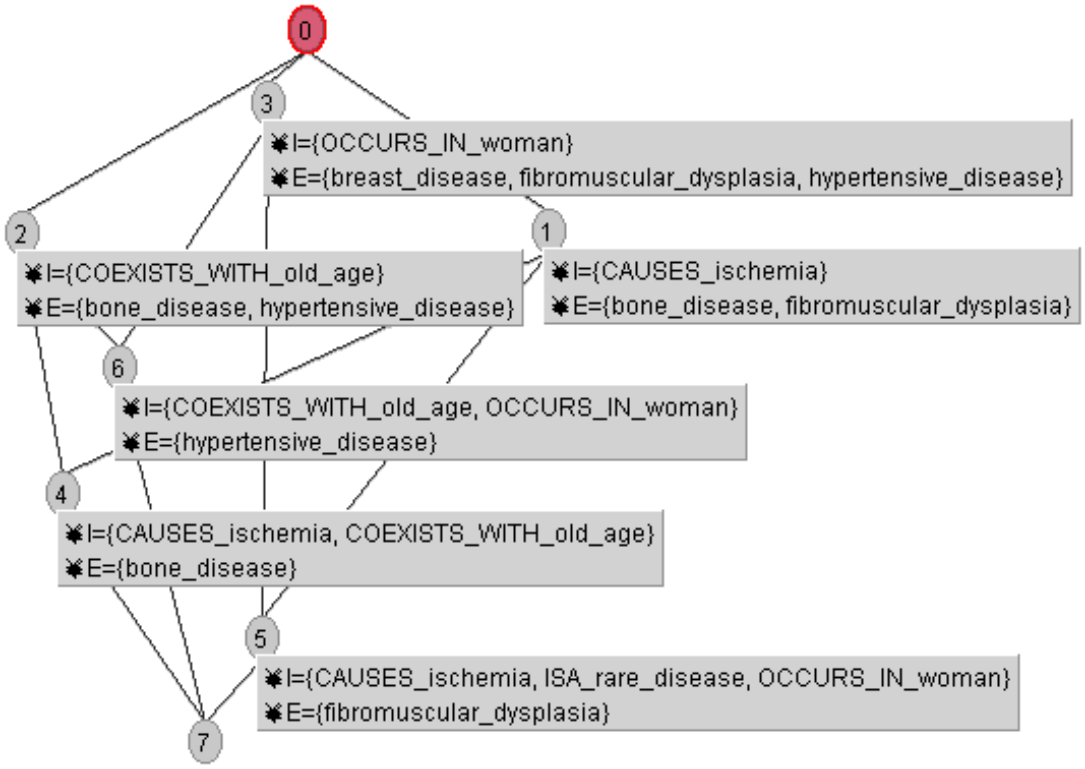

Figure 3.7: The lattice $\mathcal{L}$.

changes can be applied. And finally, we show how the formal context, the concept lattice and the annotations get evolution according to the changes (step 6 and step 7 in Figure 3.2).

3.4.1 Formulating the Changes

To meet the requirement of changes for expert interaction as described in feature 2 in section 3.2.1, we provide domain experts with three levels of changes in KESAM:

- **Changes in the formal context** for refining the domain and the scope,
- **Changes in the concept lattice** for refining the knowledge model,
- **Changes in semantic annotations** for refining the source of knowledge.

The first option that the KESAM system offers to domain experts is to make changes in the formal context for refining the domain and the scope. Table 3.3 lists the changes in KESAM that experts can do on a formal context. Experts can browse the formal context, remove some objects or attributes that are noisy or not meaningful for them or unifying objects or attributes that have the same meaning, etc. For example, there can be two objects `women` and `female` extracted from the texts. From the point of view of the expert knowledge, those objects are actually the same object, they should be unified.

The second option that the KESAM system offers to domain experts is to make changes in the concept lattice for refining the knowledge model based on the concept lattice. Experts can browse concepts in the lattice, run through subsumption paths, look at extents and intents of

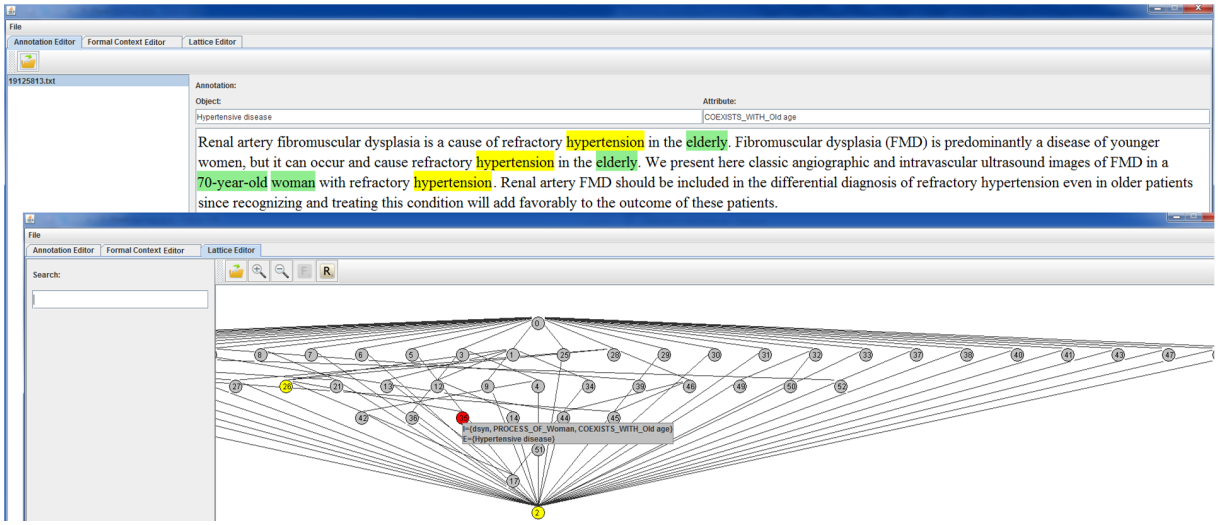

Figure 3.8: A example of the traceability in the KESAM system

Change
Add an object to the formal context
Remove an object from the formal context
Unify objects in the formal context that have the same meaning
Add an attribute to the formal context
Remove an attribute from the formal context
Unify attributes in the formal context that have the same meaning
Add an attribute to an object
Remove an attribute from an object

Table 3.3: Changes in a formal context.

concepts. Then, they can express their wishes of changes in the lattice. For example, experts evaluate the lattice given in Figure 3.7, if they are not pleased with concept C_6, then they can ask for removing this concept from the lattice. In order to define the set of changes in the lattice, we have been inspired by researches on ontology evolution which establish the taxonomies of changes for the given ontology models [Stojanovic, 2004, Klein, 2004, Luong, 2007] and adapted them to the concept lattice in FCA. Table 3.4 lists the basic changes on a lattice that add or remove only one element of the lattice. There could be more complex changes such as merging two concepts into one, splitting one concept into two sub-concepts, creating a common concept for a set of concepts, etc. These complex changes can be done by combining several basic changes in the lattice.

The third option that the KESAM system offers to domain experts is to make changes in semantic annotations for refining the source of knowledge. Adding background knowledge or editing the source of knowledge is an essential requirement for bridging the gap between the symbols extracted from texts and the authors' model in the process knowledge extraction from texts (see section 2.2.2). However, most of the existing approaches [Cimiano and Völker, 2005, Szulman et al., 2010, Tissaoui et al., 2011] do not provide experts a way for adding or editing background knowledge (Table 3.1). TERMINAE [Aussenac-Gilles et al., 2008] allows

Change	Add	Remove
	Add an object to a lattice	Remove an object from a lattice
	Add an object to a concept	Remove an object from a concept
	Add an attribute to a lattice	Remove an attribute from a lattice
	Add an attribute to a concept	Remove an attribute from a concept
	Add an attribute to an object	Remove an attribute from an object
	Add a concept to a lattice	Remove a concept from a lattice
	Add a super concept to a concept	Remove a super concept from a concept
	Add a sub concept to a concept	Remove a sub concept from a concept

Table 3.4: Basic changes in a lattice.

Change
Add an annotation
Remove an annotation
Update object of an annotation
Update attribute of an annotation
Update positions of an annotation
Update status of an annotation
Unify two annotations

Table 3.5: Changes in semantic annotations.

experts to add some explanations for concepts in the knowledge model, but it does not actually allow experts to add background knowledge that is used to build the knowledge model. When experts trace back from the lattice to the source of knowledge in texts, they may find that some necessary background knowledge is missing or some source of knowledge in texts is annotated in a wrong way. In this case, they may want to make changes in semantic annotations. To fulfill this requirement, KESAM provides an option for experts to make changes in the semantic annotations for refining the source of knowledge. The changes in annotations in KESAM are listed in Table 3.5.

With the three options for making changes in the KESAM system, domain experts can make changes for the refinement in the formal context, the concept lattice or semantic annotations. Any changes in the formal context or the concept lattice reflexes in semantic annotations, and vice versa. A change in the lattice is a kind of retro-engineering on the lattice: Experts select a change on the lattice, the system exploits the formal properties of the lattice to find all possible strategies that can meet the requirements, and then suggest these strategies with their consequences to experts. Once the experts have chosen a strategy for the change, the system then updates the semantic annotations according to the chosen strategy (step 6 and step 7 in Figure 3.2). In the next iteration, these semantic annotations are used to rebuild the lattice so that the lattice and its semantic annotations can meet the expert requirements.

In what follows, we present an approach for implementing the changes in the KESAM system.

3.4.2 Implementing the Changes

Once experts ask for a change in a concept lattice, the KESAM system exploits the formal properties of the lattice to determine all the possible strategies to suggest to the experts. Usually, several strategies can be applied to the lattice in order to perform the desired change and ensuring that the new structure remains a lattice. There is only one strategy for the changes in the formal context and annotations. A change strategy can lead to some other changes in the lattice. For example, removing a concept in the lattice or removing an annotation can lead to modify, delete some other concepts or create some new concepts. These induced changes are called *consequence* of a change strategy. It is important for domain experts to know a consequence of a change in the lattice because it can lead to delete some important concepts for them. Therefore, the consequence of a change strategy on the lattice is reported to the experts before it is actually applied so that the experts can decide which strategy should be applied.

To perform changes, for each change, we define an algorithm to compute all the possible strategies and their consequences, to identify related changes in the annotations (presented in [Tang and Toussaint, 2013]). Our algorithms benefit from the incremental approaches for building lattices [Godin et al., 1995, Kuznetsov and Obiedkov, 2002] not only to avoid complete recalculation of the lattice, but also for identifying the possible strategies and their consequences on the current lattice.

The new lattice $\mathcal{L}^*$ after a change is obtained from the existing lattice $\mathcal{L}$ by taking, deleting, modifying some concepts and/or creating some new concepts. We distinguish four possible categories of concepts in the concept lattice $\mathcal{L}^*$: *old* concepts, *deleted* concepts, *modified* concepts and *new* concepts.

Let C be a concept in $\mathcal{L}^*$, extent and intent of concept C be denoted by `Extent(C)` and `Intent(C)` respectively. The four categories of concepts are defined as as follows.

- C is an *old* concept if there exists a concept in $\mathcal{L}$ that has the same extent and intent to C,
- C is a *modified* concept if there exists a concept in $\mathcal{L}$ that has the same intent to `Intent(C)` but the extent is different from `Extent(C)`,
- C is a *new* concept if `Intent(C)` doesn't exist in $\mathcal{L}$,
- C in $\mathcal{L}$ is a *deleted* concept if `Intent(C)` doesn't exist in $\mathcal{L}^*$.

The new lattice $\mathcal{L}^*$ is generated from the existing lattice $\mathcal{L}$ by identifying these categories of concepts. The KESAM system provides experts with the consequences of changes that contains the information about the sets of modified, deleted, and new concepts so that experts can acknowledge the difference between lattices $\mathcal{L}$ and $\mathcal{L}^*$ and avoid deleting important concepts for them. Formally, consequence of a change in a lattice in KESAM is defined as below.

Definition 3.2 (Consequence of a change)**.** A *consequence of a change* in a lattice in the KESAM system is the sets of modified, deleted, and new concepts after applying that change to the lattice.

The algorithm calculating the new lattice is based on the fundamental property of lattices [Barbut and Monjardet, 1970] and the following proposition.

For any closed set (X, X') in the lattice:

$$X' = f(X) = \bigcap_{x \in X} f(\{x\})$$
$$\text{and } X = g(X') = \bigcap_{x' \in X'} g(\{x'\}).$$

ID	Strategies
S1	Removing attribute `OCCURS_IN_woman` from concept C_6
S2	Removing attribute `COEXISTS_WITH_old_age` from concept C_6
S3	Adding attributes `ISA_rare_disease` and `CAUSES_ischemia` to concept C_6

Table 3.6: Strategies for removing concept C_6.

Moreover, for any set of $f(x)$ (resp. $g(\{x'\})$), their intersection should be in the lattice.

Proposition 3.1. Let $(X, X'), (X_1, X_1') \in \mathcal{L}^*$. If $X_1' \subset X'$, then $X_1 \supset X$.

The main process of the algorithm is to update the set of intersections of extents. We perform a top-down traversal of the lattice $\mathcal{L}$ to identify whether a concept is *old*, *modified* or *deleted*, to create *new* concepts. Thanks to the categories of concepts, the resulting lattice can be calculated and we can keep a trace from the previous lattice. By this way, we can know the consequence of a change on the lattice (the sets of modified, deleted and new concepts).

We illustrate our algorithm on the change of moving concept from the lattice. For example, experts evaluate the lattice given in Figure. 3.7, they are not pleased with concept C_6 and ask for removing this concept from the lattice. The strategies for removing concept are suggested for refining the set of attributes in the intent of the concept, removing or adding attributes, by the help of its super-concepts or sub-concepts. Table 3.6 shows the strategies for removing concept C_6 from the initial lattice given in Fig. 3.7. Here, three strategies are suggested for refining the set of attributes in the intent of concept C_6: ($S1$) removing attribute according to the super-concept C_2; ($S2$) removing attribute according to the super-concept C_3; ($S3$) adding attributes according to the sub-concept C_7.

The set of objects and the set of properties of the formal context remain unchanged. For the strategies adding attributes to the intent of a concept according to one of its sub-concepts, C_{child}, the relation I of the formal context would be modified to I^* as follows:

$I^* = I \cup \{(g, m) : m \in \{\texttt{Intent}(\texttt{C}_{\texttt{child}}) \setminus \texttt{Intent}(\texttt{C})\}, g \in \{\texttt{Extent}(\texttt{C}) \setminus \texttt{Extent}(\texttt{C}_{\texttt{child}}), gIm\}$.

For the strategies removing attributes from the intent of a concept according to one of its super-concepts, C_{parent}, the relation I of the formal context would be modified to I^* as follows:

$I^* = I \setminus \{(g, m) : m \in \{\texttt{Intent}(\texttt{C}) \setminus \texttt{Intent}(\texttt{C}_{\texttt{parent}})\}, g \in \{\texttt{Extent}(\texttt{C})\}, gIm\}$.

In *Strategy 1* in Table 3.6, the set of objects and the set of properties of the formal context remain unchanged. Only the I relation is enriched: A new closed set $(\texttt{Extent}(\texttt{C}), \texttt{Intent}(\texttt{C}_{\texttt{child}}))$ becomes a formal concept of the new lattice $\mathcal{L}^*$. Figure. 3.9 shows an example of removing concept C_6 according to strategy $S1$, removing attribute `OCCURS_IN_woman` from concept C_6. In this example, concept C_6 in the initial lattice, is *deleted*; concept C_3 is *modified*, lost the object `hypertensive_disease` from its extent; the other concepts are *old*.

The detailed description of the algorithm is given in Algorithm 1. Lines 3-17 perform a top-down traversal of the lattice $\mathcal{L}$ to identify whether a concept is *old*, *modified* or *deleted* and to create *new* concepts. We call objects and attributes relating to the new relations are modified object and modified attributes, respectively. A concept in the initial lattice which does not contain any modified attribute or object remains unchanged and becomes an *old* concept in the new lattice. Line 4 is used to set *old* as a default for all the existing concepts. Concepts whose extent contains modified objects are stored in a first-in-last-out stack and the sets of extent intersections are updated (lines 5-10). The main process of the algorithm is to calculate the set of intersections of these extents. According to Proposition 1, a concept whose intent contains X' contains modified objects in its extent: lines 6-8 add modified objects to the extents of such

Algorithm 1 Removing a concept from a lattice

```
1: procedure RemoveConcept(In: 𝓛 a lattice, (X, X′) a new closed set; Out: 𝓛* a lattice)
2:     Local: S a first-in-last-out stack
3:     for each C ∈ 𝓛 do {in the ascending cardinality order of intents}
4:         Mark C as old;
5:         if Intent(C) ⊂ X′ or Extent(C) contains modified objects then
6:             if Intent(C) ⊂ X′ and X ⊈ Extent(C) then
7:                 C ← (Extent(C) ∪ X, Intent(C)) and mark C as modified;
8:             end if
9:             Put C to S;
10:            Update1(C, S);
11:        else
12:            if Intent(C) contains modified attributes then
13:                Update2(C, S);
14:            end if
15:        end if
16:        𝓛* ← C;
17:    end for
18:    for each C ∈ S do
19:        if C.mark = new then
20:            𝓛* ← C;
21:        end if
22:    end for
23: end procedure
```

concepts. Moreover, concepts with modified attributes in their intents can be modified in their extent; lines 12-14 update for these concepts. When a category is assigned to a concept, the concept is added to the new lattice (line 16). Finally, new concepts are added to the new lattice (lines 18-22).

Procedure Update1(Algorithm 2) updates the concepts whose extent contains modified objects. The loop from line 2 to line 27 is used for scanning the set of concepts in the stack. Concept category is identified according to Proposition 1. Lines 14-26 are used to update the queue. Moreover, procedure Update2 (Algorithm 3) is used for updating the concepts whose only intent contains modified attributes.

The complexity time of Algorithm 1 mainly depends on the consuming time of updating the set of intersections of extents and checking whether a concept is *modified*, *deleted* or *new* in procedure Update1. Assume that the number of concepts of the existing lattice is $|L|$. Since a concept in the existing lattice is *deleted* because the new image of its extent is bigger than the old one, the number of *new* concepts is less than or equal to the number of *deleted* concepts, the number of concepts of the new lattice is less than or equal to that of the existing lattice. The worst-case happens when all the concepts contain modified object. Thus, the loop for checking the queue never exceed $|L|$ times. Moreover, the inner loop for checking categories and creating new intersections also take less than $|L|$ times. Finally, the time complexity of procedure Update1 is bounded by $\mathcal{O}(|L|^2)$. As the top-down traversal and the first-in-last-out stack are used to update the intersections to avoid going to the further concepts of a concept after visiting it, the obtained complexity of the algorithm is far less than the bound.

Algorithm 2 Procedure UPDATE1

```
1: procedure UPDATE1(In, Out: C a concept, S a stack of concepts)
2:     for each C1 ∈ S do
3:         if Extent(C) = Extent(C1) then
4:             if Intent(C1) ⊆ Intent(C) then
5:                 C1 ← (Extent(C1), Intent(C)), mark C1 as old;
6:             else
7:                 Mark C as deleted and remove C out of S;
8:                 if Intent(C1) ⊂ Intent(C) then
9:                     C1 ← (Extent(C1), Intent(C) ∪ Intent(C1)), mark C1 as modified;
10:                end if
11:            end if
12:            Exit loop;
13:        end if
14:        if Extent(C) ⊂ Extent(C1), Intent(C1) ⊄ Intent(C) and ∄C2 ∈ S : Extent(C2) =
   Extent(C) then
15:            Mark C as deleted and remove C out of S;
16:            Add (Extent(C), Intent(C) ∪ Intent(C1)) as modified to S;
17:        end if
18:        if Extent(C) ⊃ Extent(C1) and Intent(C) ⊄ Intent(C1) then
19:            C1 ← (Extent(C1), Intent(C) ∪ Intent(C1));
20:        end if
21:        if ∄C2 ∈ S : Extent(C2) = Extent(C) ∩ Extent(C1) then
22:            Add (Extent(C) ∩ Extent(C1), Intent(C) ∪ Intent(C1)) to S and mark it as new;
23:        end if
24:        if ∃C2 ∈ S : Extent(C2) = Extent(C) ∩ Extent(C1) and (Intent(C) ∪ Intent(C1)) ⊈
   Intent(C2) then
25:            C2 ← (Extent(C2), Intent(C2) ∪ (Intent(C) ∪ Intent(C1));
26:        end if
27:    end for
28: end procedure
```

3.4.3 Evolution of the Context and Annotations

Once a strategy has been chosen, the formal context is updated and the change is propagated to annotations. Let us notice that only the corresponding objects and attributes in the formal context are updated, and only the annotations that correspond to these objects and attributes are updated. For instance, the experts choose strategy $S1$ (Table 3.6) for the action *removing concept* C_6, attribute `OCCURS_IN_woman` is then removed from to object `hypertensive_disease` in the formal context and the status of the corresponding annotations is updated to *invalid* accordingly. The formal context is updated as shown in Table 3.7. Figure 3.10 shows the lattice for strategy $S1$ after the data were updated. Steps 6 and 7 in the KESAM process (Figure 3.2) undertake these tasks. In this way, the lattice and semantic annotations are consistent with the evaluation of the experts.

Algorithm 3 Procedure UPDATE2

```
1: procedure UPDATE2(In, Out: C a concept, S a queue of concepts)
2:     for each C1 ∈ S do
3:         if Intent(C) = Intent(C1) then
4:             Change C, C1 to modified;
5:             if Extent(C) ⊈ Extent(C1) then
6:                 C, C1 ← (Extent(C) ∪ Extent(C1), Intent(C));
7:                 UPDATE1(C, S);
8:             else
9:                 C ← (Extent(C1), Intent(C)) and exit loop;
10:            end if
11:        end if
12:    end for
13: end procedure
```

Object	OCCURS_IN_woman	ISA_rare_disease	COEXISTS_WITH_old_age	CAUSES_ischemia
fibromuscular_dysplasia	x	x		x
breast_disease	x			
hypertensive_disease	◯		x	
bone_disease			x	x

Table 3.7: The updated context $\mathcal{K}$.

3.5 The KESAM Implementation and Case Study

The KESAM system supports domain experts in building a knowledge model based on the concept lattice from a set of texts with semantic annotations. The KESAM system is described in terms of the main components that build up its functional features in section 3.2.1. In this section, we present the KESAM user interface developed following the KESAM methodology.

3.5.1 The KESAM User Interface and Usage Scenario

As we have described, in order to get the knowledge model based on the concept lattice and semantic annotations to be more in accordance with the experts' requirements, the KESAM system enables expert interaction by introducing a set of changes in the lattice, the formal context, and semantic annotations. For this purpose, the KESAM is composed of three main components: *Lattice Editor*, *Formal Context Editor* and *Annotation Editor* as shown in Figure 3.11. The KESAM system is implemented in Java, in which we have implemented the AddIntent incremental algorithm from [Merwe et al., 2004] for the lattice construction. The history of

Statistics	*Dataset 1*	*Dataset 2*
Number of initial annotations	427	19299
Number of objects in the initial context	157	2401
Number of attributes in the initial context	62	649

Table 3.8: *Dataset 1* and *dataset 2*.

changes is preserved in the system.

A typical usage scenario for the KESAM system is as follows. Experts launch the KESAM system by specifying a corpus, i.e. a collection of annotation files. Following the process in Figure 3.2, the annotation files get processed by the system to transform into a formal context (step 2 in Figure 3.2). The formal context transformed from the set of annotations is then displayed in the *Formal Context Editor*. In this editor, experts could see the set of objects and attributes of the formal context. The *Formal Context Editor* provides users with a set of changes in the formal context as described in Table 3.3. Experts can define the set of objects and attributes of the formal context by removing some objects or attributes that are not meaningful, adding some new objects or attributes.... From the formal context, experts could generate the lattice. The concept lattice resulting from FCA (step 3 in Figure 3.2) is displayed in the *Lattice Editor*. The *Lattice Editor* displays the lattice and provided experts with the information of intents and extents of concepts. Experts could search the concepts that they care by specifying a set of objects or attributes and ask for changes. From the *Formal Context Editor* or *Lattice Editor*, experts can trace back to the annotations which are displayed in the *Annotation Editor*. The *Annotation Editor* shows to experts a list of documents related to the annotations and highlights them. The *Annotation Editor* also provides users with a set of changes in semantic annotations for refining the source of knowledge as described in Table 3.5. Experts can switch between these editors and make changes in the formal context, the lattice or the annotations to get the lattice and the annotations as close as possible with their requirements. The KESAM system is in charge of keeping the formal context, the lattice and the annotations up to date with the changes.

3.5.2 Case Study

In order to see how the KESAM system behaves in a realistic setting, we applied the KESAM system to build knowledge models based on the concept the lattice for medical domain. We conducted two datasets, *dataset 1* and *dataset 2*, by extracting abstracts from PubMed[16]. *Dataset 1* contained 10 abstracts about `Fibromuscular dysplasia of the renal arteries`. *Dataset 1* was small so that the evaluation of resulting lattice can be performed in detail. *Dataset 2* contained 284 abstracts about `Fibromuscular dysplasia of arteries` for testing the behavior of the approach on a large dataset. Table 3.8 shows the information about the initial annotations and the initial formal contexts of *dataset 1* and *dataset 2*.

The experimental results on the dataset is summarized in Table 3.9. The required time for building the set of annotations, transferring and visualizing the formal context and the processing time for updating the lattice and annotations at the time being can be acceptable. Optimizing the system's still in progress.

For the experiment on *dataset 1*, we reached the final lattice satisfying the requirements from the expert after 12 iterations. At the beginning, we obtained `427` annotations and the formal

[16] `http://www.ncbi.nlm.nih.gov/pubmed`

Statistics	*Dataset 1*	*Dataset 2*
Number of iterations	12	40
Number of the refined annotations	438	19385
Number of objects in the refined context	125	1935
Number of attributes in the refined context	47	503
Number of concepts in the refined lattice	45	312

Table 3.9: The experimental results on *dataset 1* and *dataset 2*.

ID	**PMID**	**Text**
#1	928685	*None showed any evidence of fibromuscular dysplasia of the renal veins.*

Table 3.10: An example of tracing back to texts

context with 157 objects and 62 attributes. Annotations that have the same meaning were unified. For example, annotations of object `Fibromuscular Dysplasia` and attributes, `dysn` and `ISA_Disease`, have the same meaning that stands for disease. We asked experts to refine the formal context before building the lattice. Objects and attributes that were the results of noise or not meaningful were removed, *i.e.* attribute `PROCESS_OF_Patients`, which stands for concerning patients, is not meaningful as all diseases concern patients. The traceability was helpful in finding noise. For instance, from the formal context, we looked back to the texts that mention object `Structure of renal vein` and property `LOCATION_OF_Fibromuscular Dysplasia`. According to the meaning of the text (ID #1 in Table 3.10, objects are colored in yellow, properties in green), we found that object `Structure of renal vein` (*renal veins*) should not have property `LOCATION_OF_Fibromuscular Dysplasia` (*fibromuscular dysplasia* in the text). At the first loop, the formal context contained 146 objects and 56 attributes. The lattice at the first loop contained 60 concepts. Then, experts verified in the lattice. They found some anomalies in the lattice and tried to improve it using the KESAM system. Throughout the iterations, the lattice got better progressively. We reached the final the lattice after 12 iterations. In the final knowledge model based on the concept lattice, we distinguish clearly two kinds of concepts: diseases and body parts. The group of concepts about body parts was split according to the diseases that often occur in. One of them was a concept grouping a set of artery objects (`Structure of renal artery`, `Entire renal artery`, `Carotid Arteries`, `Entire right renal artery`) as the body parts that disease `Fibromuscular Dysplasia` occurs in, which is mainly mentioned in the dataset.

Making changes in a knowledge model is quite complicated from users' point of view. It is not easy to say what changes should be done in a knowledge model. For example, it is not easy to say which attributes should be characterized for a concept. Our work uses FCA to find all the concepts and suggests to users as a guideline. The system KESAM is our first attempt of translating from the changes in the knowledge model based on the concept lattice and annotations to the language that experts can say. The number of iterations for reaching the desired knowledge model based on the concept lattice depends on the way that experts verify and make changes. A good change decision can reduce a lot of effort for the next iteration. The case study confirmed the advantages of unifying knowledge extraction and semantic annotation in the same process.

3.6 Conclusion

We have presented our methodology KESAM using Formal Concept Analysis for interactive and iterative extracting knowledge from texts. In KESAM, knowledge extraction and semantic annotation are unified into one single process to benefit both knowledge extraction and semantic annotation. Semantic annotations are used for formalizing the source of knowledge in texts and keeping the traceability between the knowledge model and the source of knowledge. Formal Concept Analysis is placed at the center of the KESAM system to build the knowledge model based on the concept lattice iteratively and ensure the link between knowledge model and semantic annotations. The novelty of KESAM as compared with the other work is, it is able to get feedback from domain experts to make explicit the changes that are needed for the formal context, the concept lattice and semantic annotations to be more in accordance with the experts' requirements. The KESAM process enables expert interaction by introducing the sets of changes in the formal context, the concept lattice and semantic annotations. Domain experts can evaluate and make changes in the formal context, the concept lattice or semantic annotations until they reach an agreement between the knowledge model and their own knowledge or requirements. Along with a change, its consequence in the lattice is reported to domain experts so that domain experts can decide if that change should be applied. The KESAM process then is in charge of keeping the formal context, the lattice and semantic annotations up to date with the changes. In such an interactive and iterative process, the system is able to keep the consistency between the knowledge model based on the concept lattice and semantic annotations, and converges towards a knowledge model close to the requirements of domain experts. Thanks to the link between annotations and the knowledge model, the traceability between the knowledge model and the source of knowledge in texts can be enabled, semantic annotations can be efficiently updated when the knowledge model changes, and conversely. Moreover, by classifying objects according to their attributes using FCA, the system can work with both atomic and defined concepts.

The contributions of the work presented in this chapter are the following:

- A methodology for interactive and iterative extracting knowledge from texts that benefit both knowledge extraction and semantic annotation: the traceability between the knowledge model and the source of knowledge in texts can be enabled; the knowledge model can be efficiently updated when semantic annotations change, and conversely; knowledge extraction and semantic annotation can work with both atomic and defined concepts.
- A formalization of semantic annotations for formalizing the source of knowledge in texts that is used to build the concept lattice by FCA and providing the traceability.
- A formulation of changes in formal contexts, concept lattices and semantic annotations.
- An incremental lattice building approach for implementing changes and managing the consequences of changes in the concept lattice.
- An implementation of the KESAM system. We experimented the approach for building knowledge bases about rare diseases. This approach are independent of the application domain and can be used in any field.

The lessons learned from this work are:

- It is not easy to explain why a consequence of a change happens to the lattice from expert knowledge's point of view, and therefore, it is not easy for domain experts to decide which

change strategy should be applied. Experts need to be assisted in the interpretation of the consequences of changes.

- Manually refining the lattice and observing the consequences of changes can overwhelm domain experts. An approach for reducing or simplifying the task of expert interaction in evaluation and refinement of the lattice is needed.

These problems are related to the usage of formal methods in an expert interaction system. Formal methods can help to manage the consequences of changes in the lattice and reduce expert interaction. However, formal methods do not give the explanations why those consequences of changes happen in the lattice from the expert knowledge's point of view. Therefore, bringing the meaning from the point of view of the expert knowledge to the results of formal methods is a necessary requirement. In the next chapter, we will discuss a new approach of lattice-based interaction for refining the lattice with respect to the requirements of domain experts, in which we will address the problems of using formal methods to manage the consequence of changes, to explain why the consequences of changes happen in the lattice from the expert knowledge's point of view.

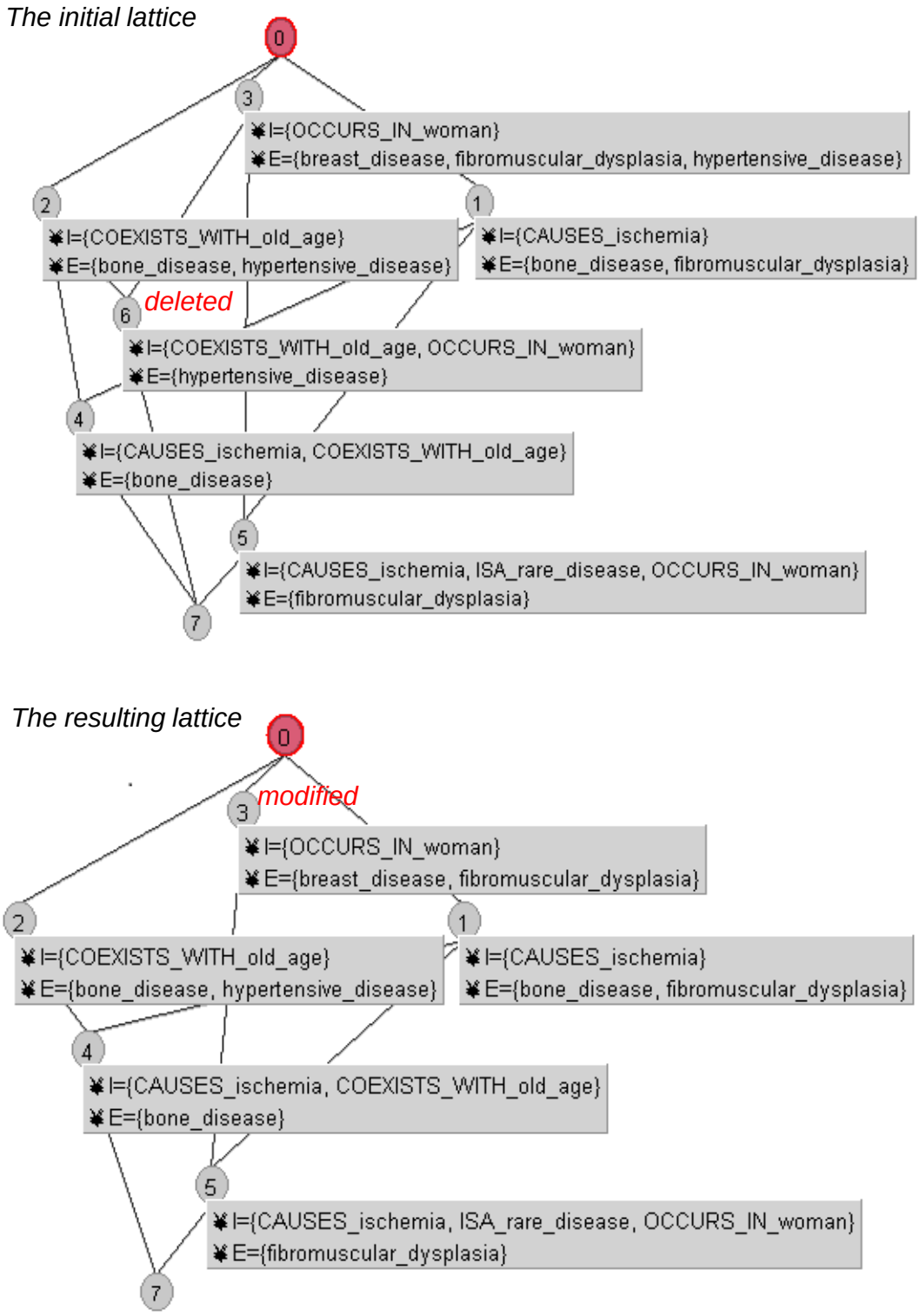

Figure 3.9: The initial lattice and the resulting lattice after removing attribute `OCCURS_IN_woman` from concept C_6 (strategy $S1$).

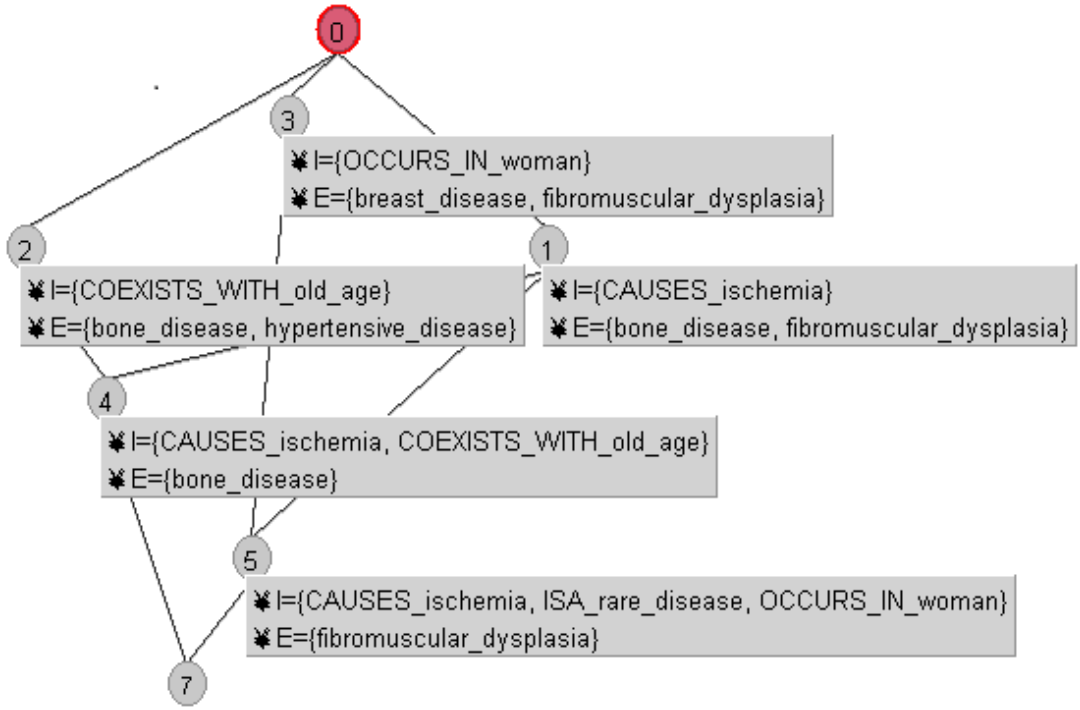

Figure 3.10: The updated lattice $\mathcal{L}$.

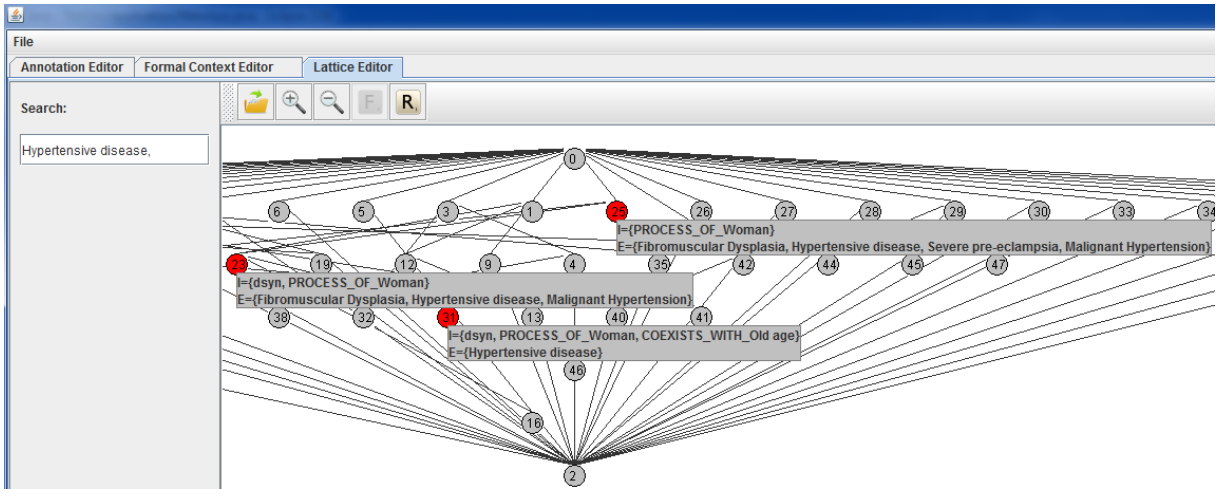

Figure 3.11: A screenshot of the KESAM system

Chapter 4

Formal Knowledge Structures and "Real-World"

Contents

4.1 Introduction

Formal Concept Analysis (FCA) [Ganter and Wille, 1999] is a formal conceptualization method which has proved to be very efficient as a bottom-up approach for building ontologies [Fennouh et al., 2014, Poelmans et al., 2013, Jia et al., 2009, Bendaoud et al., 2008a, Bendaoud et al., 2008b, Cimiano et al., 2005, Obitko et al., 2004]. FCA exploits all necessary elements for the representation of knowledge: individuals and their descriptions and elicits from data a class schema in the form of either a set of attributes implications or a concept lattice. The hierarchical structure of the concept lattice resulting from FCA fulfills the need for organizing classes into a hierarchy from the most general to the most specific concepts. The definitions of formal concepts in FCA can be used as a base for building definitions of concepts in a description logic knowledge base [Bendaoud et al., 2008a, Bendaoud et al., 2008c, Rouane-Hacene et al., 2008, Huchard et al., 2007, Rouane-Hacene et al., 2007]. The lattice structure is suitable for building the concept hierarchy of an ontology, i.e. the knowledge model of an ontology: it allows a concept to have more than one super-concepts or sub-concepts, which is close to the real-life concept organization [Ganter and Wille, 1999, Wille, 2002, Cimiano and Völker, 2005, Jia et al.,

2009]. Moreover, the logical structures of formal concepts and concept lattices are effective in supporting human reasoning [Wille, 2002, Napoli, 2005]. The lattice is thus considered as a knowledge model where domain experts could select the concepts that fit their needs to build the final ontology. However, building knowledge bases is a cognitive process and does not obey to strict and formal rules, domain experts may understand the domain in a different way than what is represented in data, and often there exists a gap between the representation model based on a concept lattice and the representation model of a domain expert. For example, in the domain of animals, an expert may expect that the rule "mammal implies do not lay eggs" holds, while this may not be the case if the platypus is among the objects in the formal context. Domain experts may not be happy with the knowledge model provided by the lattice and thus, would like to make it to be more in accordance with their own knowledge. There are several reasons that can lead to the gap between the knowledge model based on a concept lattice and the representation model of a domain expert: (1) the fact that FCA builds formal concepts from object descriptions (bottom-up approach) makes it prone to unwanted concept creation if there is noise, errors or exceptions in the data; (2) formal concepts may represent unwanted levels of granularity.

In order to bridge the possible gap between the representation model based on a concept lattice and the representation model of a domain expert, researchers [Carpineto and Romano, 2004,Belohlavek and Sklenar, 2005,Messai et al., 2008] have tried to integrate experts' knowledge in the form of dependencies between attributes into lattices. Asking for removing some attributes from the formal context as described in the KESAM system in chapter 3 would be straightforward if those attributes are noise or not meaningful for domain experts. However, this is not always the case. When classifying objects according to their attributes, humans usually follow a top-down approach and the importance of attributes is often considered: important attributes are used to form upper concepts, and less important attributes are used to form lower concepts. The relationships between the importance of attributes present "natural" dependencies which are familiar with humans in ordinary life. For example, when classifying animals according to their attributes {`is_animal`, `is_vertebrate`, `is_invertebrate`,...}, an expert has in her mind a representation model given in Figure 4.1. First, she looks for attribute `is_animal` to form upper concept `Animals`. Then, attributes `is_vertebrate` and `is_invertebrate` will come to form lower concepts `Vertebrates` and `Invertebrates`. Hence, attributes `is_vertebrates` and `is_invertebrate` are less important than attribute `is_animal`, i.e. attributes `is_vertebrates` and `is_invertebrate` depend on attribute `is_animal` (vertebrates and invertebrates are animals). In this example, to meet the expert's point of view, the dependencies between attributes should be taken into account in the concept lattice. Several approaches [Carpineto and Romano, 2004, Belohlavek and Sklenar, 2005, Messai et al., 2008] have been proposed to integrate the dependencies between attributes into lattices. In these approaches, the dependencies between attributes serve as constraints that lead to more meaningful concepts in a lattice: formal concepts which satisfy the constraints are then provided to experts, and formal concepts which do not satisfy the constraints are disregarded. The advantage of these approaches is to provide domain experts with more comprehensible structures of formal concepts.

Accordingly, in this chapter, we introduce an approach to bridge the possible gap between the representation model based on a concept lattice and the representation model of a domain expert. As the logical structures of formal concepts and concept lattices are effective in supporting human reasoning [Wille, 2002, Napoli, 2005], we would like to provide a formal framework for integrating expert knowledge into concept lattices in such a way that we can maintain the lattice structure. Moreover, instead of providing only concepts that satisfy experts' knowledge, the framework allows experts to keep a trace of changes occurring in the original lattice and the final constrained version. On the one side, there is a representation model based on a concept lattice

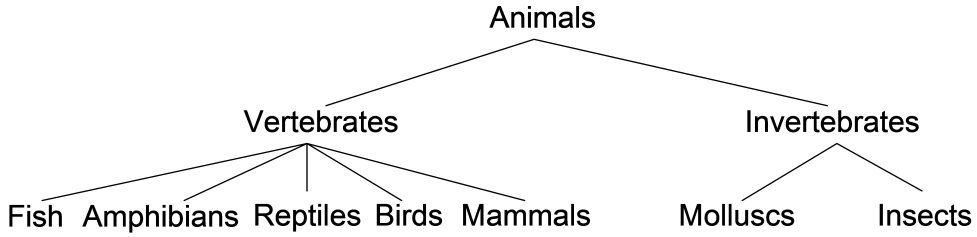

Figure 4.1: An example of animal classification.

that is automatically issued from data. On the other side, experts have a representation model in their mind. Through the trace of changes, experts can access how concepts in practice are related to concepts automatically issued from data.

In this work, in order to bridge the gap above, we "align" a set of attribute dependencies with the set of implications provided by the concept lattice, leading to modifications in the original lattice. The method extends the definition of dependencies between single attributes introduced in [Belohlavek and Sklenar, 2005] to the case of dependencies between attribute sets, and allows domain experts to have more possibilities for expressing constraints. We are able to build the constrained lattices and provide the trace of changes by using extensional projections [Ganter and Kuznetsov, 2001, Pernelle et al., 2002] over lattices. From an original lattice, two different projections produce two different constrained lattices, and thus, the gap between the representation model based on a concept lattice and the representation model of a domain expert is filled with projections.

This chapter is organized as follows. Firstly, we introduce some basic notions of attribute implications, attribute dependencies, and projections that provide the foundations of our work. Next, we detail the approach for generating constrained lattices, providing the trace of changes by using extensional projections. Finally, we conclude our work and draw some perspectives over the approach.

4.2 Preliminaries

Let us recall the definitions of attribute implications in FCA [Ganter and Wille, 1999], attribute dependencies and concept lattices constrained by attribute dependencies introduced in [Belohlavek and Sklenar, 2005, Messai et al., 2008]. Later, we will extend these definitions to deal with dependence relations between attribute sets. Examples are also given to show how this kind of knowledge is integrated into concept lattices. In addition, we examine the relation between attribute dependencies and attribute implications. The formal properties of attribute implications in lattices are then exploited to build the projections for our framework. Finally, we recall the definition of projections [Ganter and Kuznetsov, 2001, Soldano and Ventos, 2011] in a concept lattice that we will use in our contributions.

4.2.1 Attribute Implication

Implications in a formal context represent dependence relations between attributes existing in data [Duquenne and Guigues, 1986, Ganter and Wille, 1999]. In the following, we give the definition of attribute implications based on [Ganter and Wille, 1999].

Animal	has_two_legs (m1)	lays_eggs (m2)	can_fly (m3)	has_wings (m4)	has_fins (m5)	has_feathers (m6)	has_milk (m7)	has_backbone (m8)	lives_in_ water (m9)
bear (g1)	x						x	x	
carp (g2)		x			x			x	x
chicken (g3)	x	x	x	x		x		x	
crab (g4)		x							x
dolphin (g5)			x		x		x	x	x
honeybee (g6)		x	x	x					
penguin (g7)	x	x		x		x		x	x
wallaby (g8)	x						x	x	

Table 4.1: Formal context of animals

Definition 4.1 (Attribute Implication [Ganter and Wille, 1999])**.** An implication between sets of attributes $X, Y \subseteq M$ in a formal context (G, M, I) is denoted by $X \to Y$, where every object having all the attributes from X has also all the attributes from Y, i.e. $X' \subseteq Y'$.

Example 4.1. Consider a formal context given in Table 4.1. This formal context represents data about animals (g1: bear, g2: carp, g3: chicken, g4: crab, g5: dolphin, g6: honeybee, g7: penguin, g8: wallaby) and their attributes (m1: has_two_legs, m2: lays_eggs, m3: can_fly, m4: has_wings, m5: has_fins, m6: has_feathers, m7: has_milk, m8: has_backbone, m9: lives_in_water). In this formal context, implication m6 : has_feathers $\to$ m8 : has_backbone holds because every object having attribute m6: has_feathers has also attribute m8: has_backbone;
implication {m5 : has_fins} $\to$ {m8 : has_backbone, m9 : lives_in_water} holds because every object having attribute m5:has_fins has also all the attributes from the attribute set {m8: has_backbone, m9:lives_in_water}.

An attribute implication can be read off from a formal context by Proposition 4.1 or from a concept lattice by Proposition 4.2 [Ganter and Wille, 1999].

Proposition 4.1 ([Ganter and Wille, 1999])**.** An implication $X \to Y$ between set of attributes $X, Y \subseteq M$ holds in (G, M, I) iff $Y \subseteq X''$, where X' is the set of objects which have all the attributes in X, X'' standing for $(X')'$ is the set of attributes which are common to all the objects in X'. It then automatically holds in the set of all concept intents as well.

Proposition 4.2 ([Ganter and Wille, 1999])**.** An implication $X \to Y$ between set of attributes $X, Y \subseteq M$ holds in a lattice iff $X \to m$ holds for each $m \in Y$. $X \to m$ holds iff $(m', m'') \geq (X', X'')$, where X' is the set of objects which have all the attributes in X, X'' standing for $(X')'$ is the set of attributes which are common to all the objects in X', m' is the set of objects which have attribute m, m'' standing for $(m')'$ is the set of attributes which are common to all the objects in m'. (m', m'') is the attribute concept of m.

For each formal context, there exists a sound and complete set of attribute implications called *implication base* or *Duquenne-Guigues base* [Duquenne and Guigues, 1986].

4.2.2 Constrained Concept Lattices w.r.t. Attribute Dependencies

Different from implications, attribute dependencies do not arise from data. They are dependence relations that experts *expect* to exists as attribute implications. Attribute dependencies represent experts' knowledge when classifying objects according to their attributes such that, important attributes are used to form upper concepts, and less important attributes are used to form lower concepts. Attribute dependency was firstly introduced by Belohlavek and Sklenar [Belohlavek and Sklenar, 2005] and then extended by Messai et al. [Messai et al., 2008] as a formalization of dependency relations between single attributes. In the following, we provide the definition of attribute dependencies based on [Belohlavek and Sklenar, 2005] and [Messai et al., 2008].

Definition 4.2 (Attribute Dependency adapted from [Messai et al., 2008])**.** An *attribute dependency*, denoted as $x \prec y$, is a representation of an expert's belief, where all objects having attribute x *should* also have attribute y. Consider an attribute dependency as a simile to an attribute implication which does not necessarily hold in data.

Example 4.2. Consider the dataset about animals depicted in the formal context shown in Table 4.1. If an expert believes all animals that have wings can fly, we can represent this belief as `m4 : has_wings` $\prec$ `m3 : can_fly`.

Clearly, this belief is wrong as penguins cannot fly and have wings. Thus, there is no implication in the form `m4 : has_wings` $\rightarrow$ `m3 : can_fly` that hold in the formal context. In the following, we will show how to deal with this situation.

Definition 4.3 (Formal Concept Satisfaction [Belohlavek and Sklenar, 2005])**.** A formal concept (A, B) satisfies an attribute dependency $x \prec y$ between attributes x and y iff whenever $x \in B$ then $y \in B$.

Example 4.3. Consider the concept lattice built from the formal context given in Table 4.1 as shown in Figure 4.2. In this lattice, concept C_{12} whose intent is {`m2: lays_eggs`, `m3: can_fly`, `m4: has_wings`} satisfies attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly` because its intent contains both attributes `m4 : has_wings` and `m3 : can_fly`; concept C_4 whose intent is {`m3: can_fly` } satisfies attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly` because its intent does not contain attribute `m4 : has_wings`. Instead, concept C_{11} whose intent is {`m2 : lays_eggs`, `m4 : has_wings`} does not satisfy attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly` because its intent contains attribute `m4 : has_wings`, but not attribute `m3 : can_fly`.

Definition 4.4 (Constrained Poset [Belohlavek and Sklenar, 2005])**.**

1) A concept lattice $\mathcal{L}$ constrained by an attribute dependency $x \prec y$, is the collection of all formal concepts from lattice $\mathcal{L}$ which satisfy $x \prec y$.

2) A concept lattice $\mathcal{L}$ constrained by a set of attribute dependencies D, is the collection of all formal concepts from lattice $\mathcal{L}$ which satisfy all attribute dependencies in D.

Notice that both collections are partially ordered subsets of the original lattice $\mathcal{L}$ [Belohlavek and Sklenar, 2005]. We will refer to these collections as *constrained posets of lattice $\mathcal{L}$ w.r.t. dependency $x \prec y$ (or w.r.t. the set of dependencies D).*

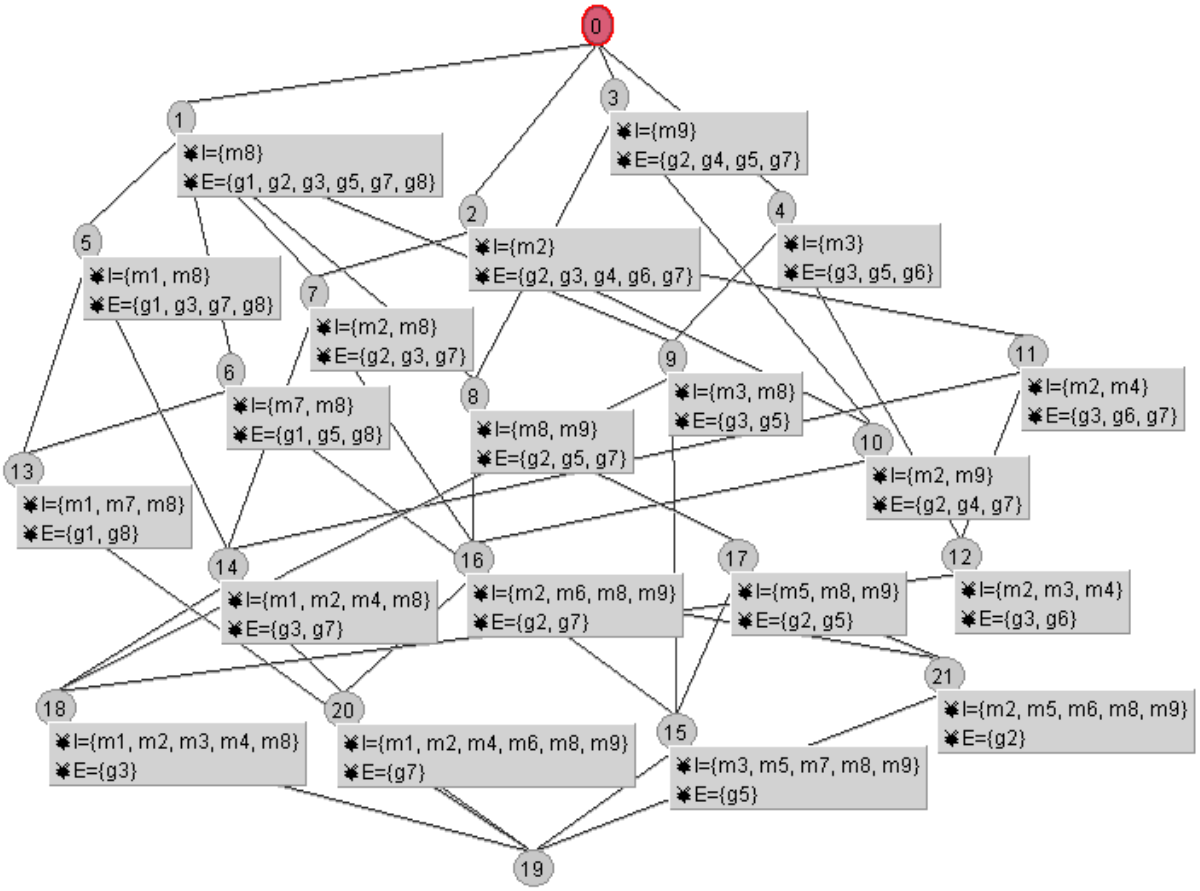

Figure 4.2: The concept lattice built from the formal context given in Table 4.1.

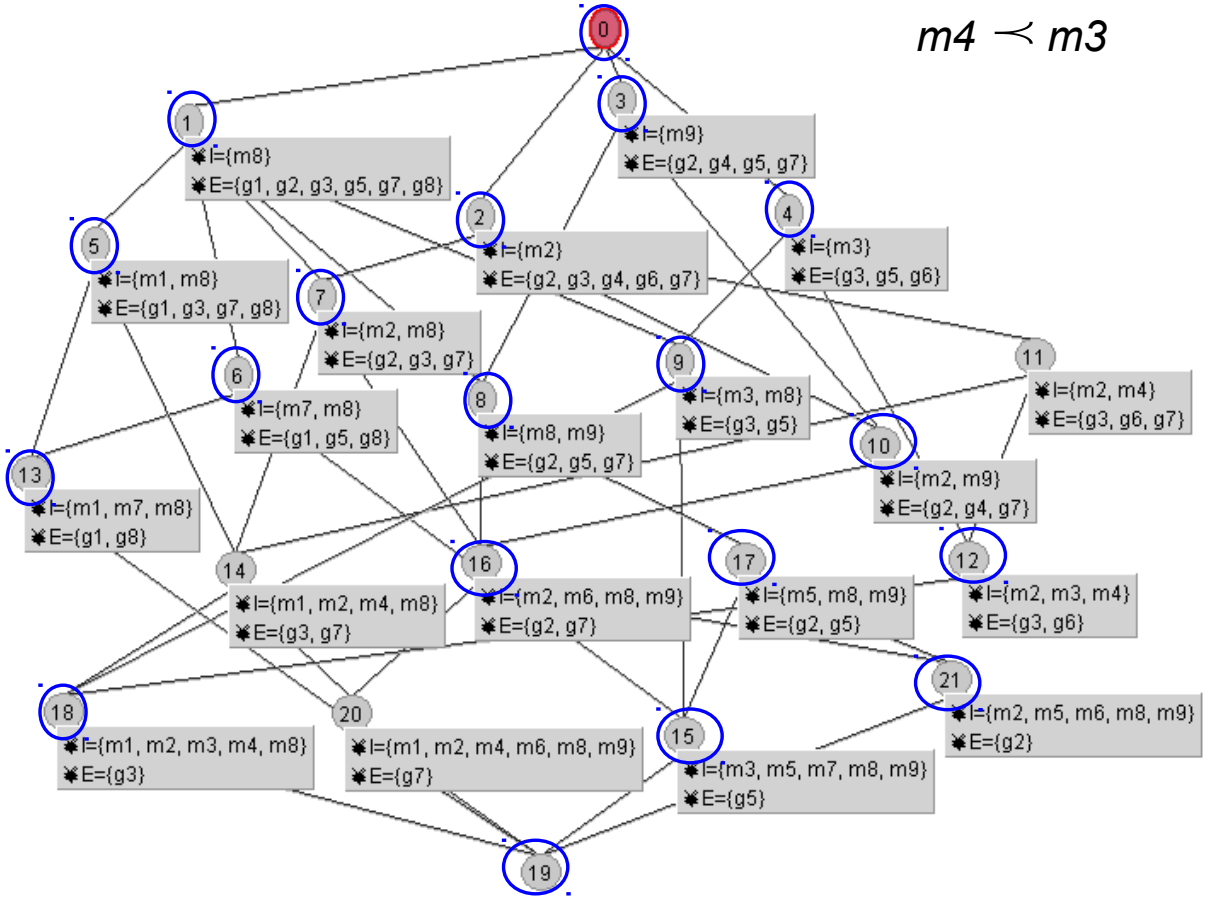

Figure 4.3: The lattice constrained by `m4` ≺ `m3`.

Example 4.4. Consider the dataset about animals, i.e. the formal context given in Table 4.1. From the formal context, we get the concept lattice as shown in Figure 4.2. Suppose that an expert provides her knowledge in the form of an attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly`. This attribute dependency serves as a constraint to get formal concepts which satisfy the expert's point of view. The constrained poset of $\mathcal{L}$ by attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly` is the collection of all formal concepts from the lattice shown in Figure 4.2 which satisfy this attribute dependency, i.e. the collection of the formal concepts circled blue in Figure 4.3. Concepts C_{11}, C_{14}, C_{20} in Figure 4.3 which do not satisfy attribute dependency `m4:has_wings` $\prec$ `m3 : can_fly` are disregarded.

Both attribute implications and attribute dependencies represent the dependencies between attributes. Attribute implications come out from data while attribute dependencies encode experts' knowledge. Experts may say things different than data, and thus, an attribute dependency may not fit with any implication extracted from data. This is what we call the gap between gap between the representation model based on a concept lattice and the representation model of a domain expert.

Example 4.5. Consider again the data about animals, the formal context given in Table 4.1. Suppose that an expert wants an attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly`. This attribute dependency does not fit any implication extracted from the data, i.e. `m4 : has_wings` $\rightarrow$ `m3 : can_fly` is not an actual implication. In this situation, attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly` represents an "expertise refinement".

4.2.3 Projections

Projections are mathematical functions that allow simplifying a concept lattice by mapping formal concepts through functions applied to the extent or the intent (*extensional* or *intensional* projections respectively [Pernelle et al., 2002, Soldano and Ventos, 2011]). Projections are also used for simplifying descriptions of concepts in pattern structures [Ganter and Kuznetsov, 2001, Buzmakov et al., 2015]. In partial order theory, projections are known as *kernel operators* or *interior operators*. For our purpose, we use extensional projections.

Definition 4.5 (Extensional Projection adapted from [Pernelle et al., 2002])**.** ψ is an *extensional projection* of a lattice $\mathcal{L}$ iff for each pair (A_1, A_2) of extents of $\mathcal{L}$, we have:

- if $A_1 \subseteq A_2$, then $\psi(A_1) \subseteq \psi(A_2)$ (monotone),
- $\psi(A_1) \subseteq A_1$ (contractive), and
- $\psi(\psi(A_1)) = \psi(A_1)$ (idempotent).

The result of a projection over a lattice is also a lattice as explained in Proposition 4.3 [Pernelle et al., 2002].

Proposition 4.3 ([Pernelle et al., 2002])**.** Let $\mathcal{L}$ be a lattice with the join operator $\vee_{\mathcal{L}}$ and meet operator $\wedge_{\mathcal{L}}$, and ψ be an extensional projection of $\mathcal{L}$, then $\psi(\mathcal{L})$ is also a lattice with the join operator $\vee$ and the meet operator $\wedge$ defined as, for any pair $A_1, A_2 \in \psi(\mathcal{L})$:

- $A_1 \vee A_2 = A_1 \vee_{\mathcal{L}} A_2$
- $A_1 \wedge A_2 = \psi(A_1 \wedge_{\mathcal{L}} A_2)$

Let $\mathcal{L}$ be a lattice and ψ an extensional projection of $\mathcal{L}$, then the set of extents E in $\mathcal{L}$ can be divided into two sets:

$E = \{e \in E | \psi(e) = e\} \cup \{e \in E | \psi(e) \neq e\}$. The set $\{e \in E | \psi(e) = e\}$ is called the *fixed point* of ψ.

The mapping of a concept in a lattice onto the corresponding concept in the projected lattice can be computed thanks to Proposition 4.4.

Proposition 4.4 ([Soldano and Ventos, 2011])**.** Let $\mathcal{L}$ be a lattice and ψ be an extensional projection of $\mathcal{L}$, then a concept (A, B) in $\mathcal{L}$ is projected in the corresponding lattice $\psi(\mathcal{L})$ on the concept (A_1, B_1) such that $A_1 = \psi(A)$ and $B_1 = A_1'$.

It is worth noticing that the result of a projection is actually a concept lattice. For our purposes, this adds the benefit that the result of constraining a lattice through a projection preserves the lattice structure. Hereafter, we will refer to the result of constraining the lattice through a projection as a *constrained lattice*.

4.3 Projections for Generating Constrained Lattices

As previously discussed, in a given formal context, there exists some implications that represent attribute dependencies among attributes. However, from the perspective of a domain expert some implications may not be presented in data for different reasons. For example, some objects may have missing attribute associations or may have wrongly assigned attributes. In a different scenario, an expert may simply want to observe formal concepts aligned through her particular point-of-view of the domain. Thus, often there exists a *gap* between the relations of attributes in the data and the relations of attributes as a domain expert understands them.

4.3.1 Discussion about Constrained Lattices

In order to bridge the gap between the representation model based on a concept lattice and the representation model of a domain expert, we "align" a set of attribute dependencies with the set of implications provided by the concept lattice. According to Definition 4.2, if an implication $x \rightarrow y$ holds in a lattice, then that lattice satisfies the attribute dependency $x \prec y$. To build a lattice satisfying a set of attribute dependencies, we look for a lattice that holds the corresponding set of attribute implications. In our setting, we want to use a well-founded process base on projections. Thus, we provide a method for constraining the lattice in such a way that the lattice structure is preserved. Moreover, we provide experts with explanations on why some concepts in the original lattice disappear. The explanations are provided in the form of a mapping of these concepts onto the corresponding concepts in the constrained version. We refer to these mappings as the *trace of changes* occurring in the original lattice and the final constrained version. We achieve this by using extensional projections over lattices.

We illustrate this scenario as shown in Figure 4.4, where a lattice $\mathcal{L}$ is mapped onto a lattice $\mathcal{L}_1$ which is a constrained version w.r.t. the attribute dependency $x \prec y$. Let us call this mapping ς.

We observe the following characteristics of ς:

(i) ς reduces the size of the lattice $\mathcal{L}$ because the constrained lattice $\mathcal{L}_1$ do not contain the formal concepts in $\mathcal{L}$ that do not satisfy the attribute dependency $x \prec y$,

(ii) According to the lattice structure, ς reduces the concept extents while increasing their intents:

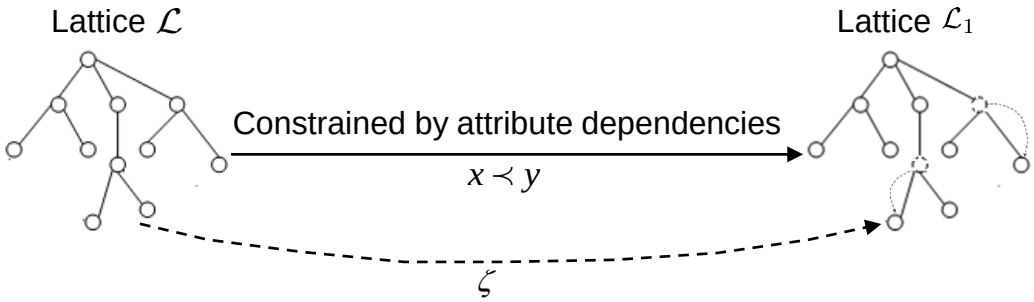

Figure 4.4: Mapping function between a lattice and the lattice constrained by attribute dependencies.

- If a concept $C \in \mathcal{L}$ satisfies attribute dependency $x \prec y$, then $\varsigma(C) = C$.
- If a concept $C \in \mathcal{L}$ does not satisfy $x \prec y$ (the intent of C contains x, but not y), let us denote intent of a concept C by $int(C)$, then, in order to satisfy $x \prec y$ in $\mathcal{L}_1$, $int(\varsigma(C)) = int(C) \cup K$, where K is a set of attributes that contains y.

By this observation, we know that ς simplifies concept extents in the original lattice $\mathcal{L}$ in such a way that the size of $\mathcal{L}$ is reduced and the lattice structure is preserved. In order to get formal concepts in $\mathcal{L}$ satisfying the constraints, ς replaces the concept extents in that lattice with smaller sets of objects which are still extents. This replacement may result in a loss of information. Hence, the trace of changes is useful for domain experts to be aware that some concepts in the lattice will be lost some important objects. Indeed, (i) and (ii) are consequences of the fact that ς is an extensional projection. ς is a special case of projections from [Ganter and Kuznetsov, 2001, Pernelle et al., 2002]. This extensional projection does not create new extents, it replaces the concept extents in the lattice with smaller extents.

In the following, we describe how the extensional projection is defined in two different cases, namely for a single attribute dependency and for a set of attribute dependencies.

4.3.2 Projections for Constrained Lattices w.r.t. Dependencies between Attribute Sets

In this section, we first define the extensional projection for a simple situation of constraining the lattice w.r.t. a dependency between single attributes, and then extend the projection to the situation of constraining the lattice by a dependency between attribute sets.

Projections for Constrained Lattices w.r.t. Dependencies between Single Attributes

Let us consider the problem of finding an extensional projection $\psi : \mathcal{L} \to \mathcal{L}_1$, where $\mathcal{L}$ is a concept lattice which does not satisfy the attribute implication $x \to y$ between attributes $x, y \in M$, $\mathcal{L}_1$ is the projected lattice of $\mathcal{L}$ which satisfies the implication $x \to y$.

The following propositions state the main properties of the extensional projection ψ.

Proposition 4.5. Let $\mathcal{L}$ be a concept lattice which does not satisfy the attribute implication $x \to y$ between attributes x and y, then:
$x' \not\subseteq y' \Longrightarrow x' \cap y' \subset x'$

Proof. Because the lattice $\mathcal{L}$ does not satisfy the implication $x \to y$, according to Proposition 4.2 of attribute implications in a concept lattice, $(y', y'') \not\geq (x', x'')$. So, $x' \not\subseteq y' \Longrightarrow x' \cap y' \subset x'$.

□

Proposition 4.6. Let $\mathcal{L}$ be a concept lattice which does not satisfy the attribute implication $x \to y$ between attributes x and y, and ψ be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies $x \to y$, then:

1) $\psi(x')$ is the maximal set of objects having attribute x in the projected lattice,

2) $\psi(y')$ is the maximal set of objects having attribute y in the projected lattice, and

3) $\psi(x') \subseteq \psi(y')$.

Proof.

1) Let A be an extent in $\mathcal{L}$ such that $x \in A'$, then $A \subseteq x'$ because x' is the maximal set of objects having attribute x in $\mathcal{L}$. By the monotonic property of projections, in the projected lattice, we have $\psi(A) \subseteq \psi(x')$. So, $\psi(x')$ is the maximal set of objects having attribute x in the projected lattice.

2) Similarly, $\psi(y')$ is the maximal set of objects having attribute y in the projected lattice.

3) Because the projected lattice satisfies the implication $x \to y$, according to Proposition 4.2 of attribute implications in a concept lattice, we have $(\psi(y'), \psi(y')') \geq (\psi(x'), \psi(x')')$. So, we have $\psi(x') \subseteq \psi(y')$.

□

Proposition 4.7. Let $\mathcal{L}$ be a concept lattice which does not satisfy the attribute implication $x \to y$ between attributes x and y, and ψ be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies $x \to y$, then:
$\psi(x') = x' \cap y'$

Proof.

1) As the projected lattice satisfies $x \to y$, according to Proposition 4.6, we have $\psi(x') \subseteq \psi(y')$.

2) $\psi(y') \subseteq y'$ (by the contractive property of projections).

3) As a result of 1) and 2), we have $\psi(x') \subseteq \psi(y') \subseteq y'$.

4) $\psi(x') \subseteq x'$ (by the contractive property of projections).

5) As a result of 3) and 4), we have $\psi(x') \subseteq x' \cap y'$.

6) According to Proposition 4.6, $\psi(x')$ is the maximal set of objects having attribute x in the projected lattice $\mathcal{L}_1$.

7) We know that $x' \cap y'$ is an extent in $\mathcal{L}$, and the objects in this set have both attributes x and y, i.e. the concept with the extent $x' \cap y'$ in $\mathcal{L}$ satisfies the attribute implication $x \to y$ and remains the same in $\mathcal{L}_1$. So, the extent $x' \cap y'$ exists in $\mathcal{L}_1$ and the objects in this set contain attribute x.

8) As a result of 6) and 7), we have $x' \cap y' \subseteq \psi(x')$.

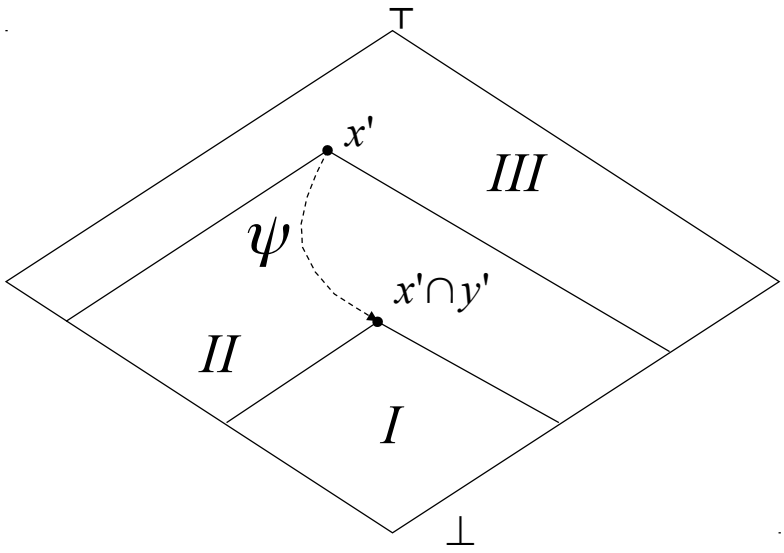

Figure 4.5: Three possible categories of extents in lattice $\mathcal{L}$ when $\psi(x') = x' \cap y'$.

From 5) and 8), we have $\psi(x') = x' \cap y'$.

□

Proposition 4.7 gives us an important property of the extensional projection ψ to observe changes in the lattice $\mathcal{L}$: From lattice $\mathcal{L}$ which does not satisfy the implication $x \to y$, we are going to project x' in lattice $\mathcal{L}$ to $x' \cap y'$ to get the projected lattice that satisfies the implication $x \to y$.

Given a concept lattice $\mathcal{L}$ which does not satisfy the attribute implication $x \to y$ between attributes x and y, an extensional projection ψ of $\mathcal{L}$ such that $\psi(x') = x' \cap y'$, extents A in lattice $\mathcal{L}$ can be divided into three categories as shown in Figure 4.5.

- Category I contains all extents A that are subsumed by $x' \cap y'$, i.e. $A \subseteq (x' \cap y') \subset x'$ ($x' \cap y' \subset x'$ by Proposition 4.5).
- Category II contains all extents A that are subsumed by x' but not subsumed by $x' \cap y'$, i.e. $A \subseteq x'$, $A \not\subseteq (x' \cap y')$.
- Category III contains extents A that are not in parts I, II, i.e. $A \not\subseteq x'$.

Consider an element A in Category I ($A \subseteq (x' \cap y') \subset x'$). $x' \cap y'$ is an extent in $\mathcal{L}$, and the objects in this set have both attributes x and y, i.e. the concept with extent $x' \cap y'$ satisfies the attribute implication $x \to y$. Because A is subsumed by $x' \cap y'$, the concept with extent A satisfies the attribute implication $x \to y$. So, the concept with extent A in $\mathcal{L}$ remains the same in the projected lattice, i.e. $\psi(A) = A$. Part I is a component of the fixed point of the projection ψ.

Consider an element A in Category III ($A \not\subseteq x'$). Because the objects in A do not have attribute x, the concept with extent A in $\mathcal{L}$ satisfies the attribute implication $x \to y$. These concepts remain the same in the projected lattice, i.e. $\psi(A) = A$. Part III is also a component of the fixed point of the projection ψ.

Consider an element A in Category II ($A \subseteq x'$, $A \not\subseteq (x' \cap y')$). We have:

1) $\psi(A) \subseteq A$ (by the contractive property of projections)

2) Because $A \subseteq x'$, $\psi(A) \subseteq \psi(x')$ (by the monotonic property of projections). Moreover, $\psi(x') = x' \cap y'$ (by Proposition 4.7). So, $\psi(A) \subseteq x' \cap y'$.

As a result of 1) and 2), $\psi(A) \subseteq A \cap (x' \cap y')$.

In order to have the largest fixed point, we set $\psi(A) = A \cap (x' \cap y')$. $\psi(A) = A \cap (x' \cap y')$ complies with the properties of projections (see Appendix) and concept with extent $\psi(A)$ satisfies the implication $x \rightarrow y$ because objects in $\psi(A) = A \cap (x' \cap y')$ have both attributes x and y. This introduces the fact that Part II constrains concepts which are projected in such a way that $\psi(A) = A \cap (x' \cap y')$.

Thus, the extensional projection with the largest fixed point among the projections given by $\psi(x') = x' \cap y'$ is:

$$\psi(A) = \begin{cases} A \cap (x' \cap y') & \text{if } A \subseteq x' \text{AND } A \not\subseteq (x' \cap y'), \\ A & \text{otherwise.} \end{cases} \tag{4.1}$$

This projection gives the projected lattice that satisfies the attribute implication $x \rightarrow y$. The trace of changes occurring in the original lattice $\mathcal{L}$ and the constrained lattice $\mathcal{L}_1$ can be obtained thanks to Proposition 4.4.

Example 4.6. Let us consider the previous Example 4.4 where the formal context is given in Table 4.1, the lattice built from this formal context shown in Figure 4.2, and the expert provides her knowledge in the form of an attribute dependency `m4 : has_wings` $\prec$ `m3 : can_fly`. According to the data in Table 4.1:

- `m4`$'$=`{g3, g6, g7}`,
- `m3`$'$=`{g3, g5, g6}`,
- `m4`$' \cap$ `m3`$'$=`{g3, g6}`.

Applying Equation 4.1, the extensional projection ψ for generating the lattice constrained by the dependency `m4 : has_wings` $\prec$ `m3 : can_fly` from the original lattice is:

$$\psi(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g6, g7}\} \text{AND } A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.2}$$

Figure 4.6 depicts the constrained lattice and the trace of changes provided by the projection ψ. In Figure 4.6, the formal concepts of the constrained lattice are circled blue. Thanks to Proposition 4.4, we obtain the trace of changes occurring in the original lattice $\mathcal{L}$ and the constrained lattice $\mathcal{L}_1$ including C_{11} in lattice $\mathcal{L}$ is changed to C_{12} in constrained lattice $\mathcal{L}_1$, C_{14} in $\mathcal{L}$ is changed to C_{18} in $\mathcal{L}_1$, and C_{20} in $\mathcal{L}$ is changed to C_{19} in $\mathcal{L}_1$.

An interpretation of the change C_{11} in $\mathcal{L}$ to C_{12} in $\mathcal{L}_1$ according to the semantics of the extensional projection is as follows. According to the data, objects `g3`: `chicken`, `g6`: `honeybee` are grouped together with object `g7`: `penguin` to form a concept C_{11} whose intent is {`m2`: `lays_eggs`, `m4`: `has_wings`}. According the expert, animals that have wings should also fly (`m4 : has_wings` $\prec$ `m3 : can_fly`), object `g7`: `penguin` should not be grouped together with objects `g3`: `chicken` and `g6`: `honeybee` to form a concept. This is better represented by concept C_{12} whose extent is {`g3`: `chicken`, `g6`: `honeybee`} and intent is {`m2`: `lays_eggs`, `m3`: `can_fly`, `m4`: `has_wings`}. It is similar for the interpretations of the changes, C_{14} in $\mathcal{L}$ is changed to C_{18} in $\mathcal{L}_1$ and C_{20} in $\mathcal{L}$ is changed to C_{19} in $\mathcal{L}_1$.

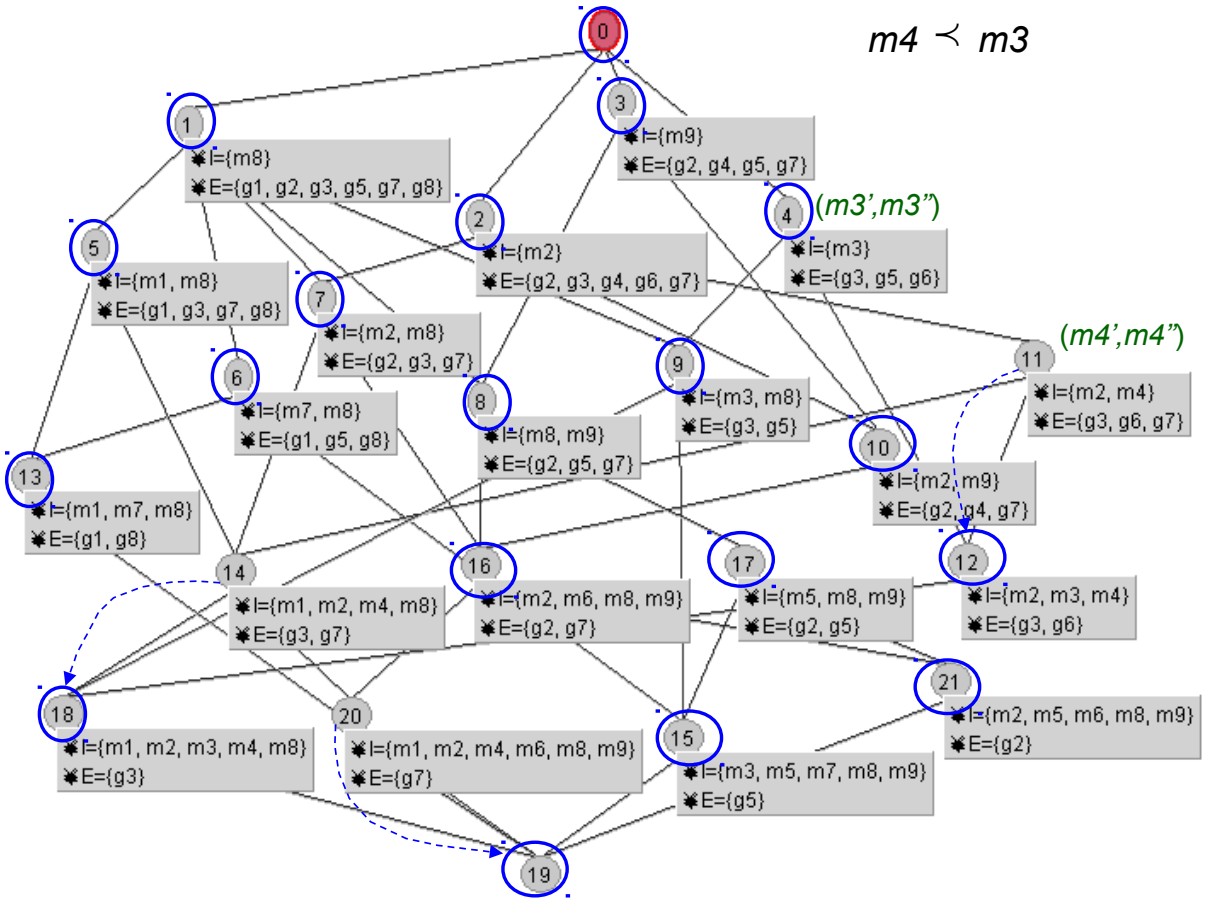

Figure 4.6: The lattice constrained by `m4` ≺ `m3` and the trace of changes.

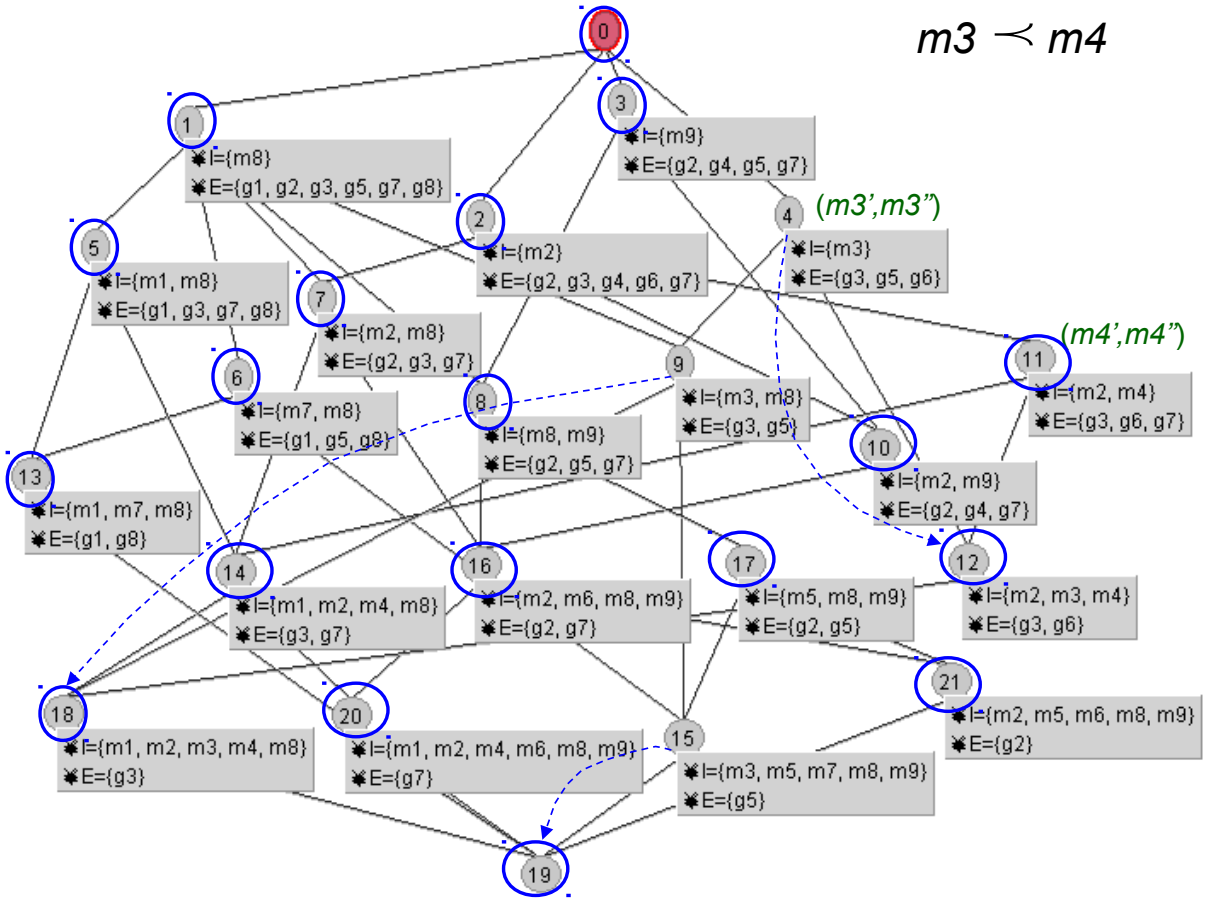

Figure 4.7: The lattice constrained by $\texttt{m3} \prec \texttt{m4}$ and the trace of changes.

Example 4.7. Consider the running example where the expert provides her knowledge in the form of an attribute dependency $\texttt{m3} : \texttt{can\_fly} \prec \texttt{m4} : \texttt{has\_wings}$. According to the data in Table 4.1:

- $\texttt{m3}' = \{\texttt{g3}, \texttt{g5}, \texttt{g6}\}$,
- $\texttt{m4}' = \{\texttt{g3}, \texttt{g6}, \texttt{g7}\}$,
- $\texttt{m4}' \cap \texttt{m3}' = \{\texttt{g3}, \texttt{g6}\}$.

Applying Equation 4.1, the extensional projection ψ for generating the lattice constrained by the dependency $\texttt{m3} : \texttt{can\_fly} \prec \texttt{m4} : \texttt{has\_wings}$ from the original lattice is:

$$\psi(A) = \begin{cases} A \cap \{\texttt{g3}, \texttt{g6}\} & \text{if } A \subseteq \{\texttt{g3}, \texttt{g5}, \texttt{g6}\}, A \not\subseteq \{\texttt{g3}, \texttt{g6}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.3}$$

Figure 4.7 depicts the constrained lattice and the trace of changes provided by the projection ψ. In Figure 4.7, the formal concepts of the constrained lattice are circled blue. Applying Proposition 4.4, the trace of changes in this example contains C_4 in lattice $\mathcal{L}$ is changed to C_{12} in constrained lattice $\mathcal{L}_1$, C_9 in $\mathcal{L}$ is changed to C_{18} in $\mathcal{L}_1$, and C_{15} in $\mathcal{L}$ is changed to C_{19} in $\mathcal{L}_1$.

An interpretation of the change C_4 in $\mathcal{L}$ to C_{12} in $\mathcal{L}_1$ according to the semantics of the extensional projection is as follows. According to the data, objects `g3`: `chicken`, `g6`: `honeybee` are grouped together with object `g5`: `dolphin` to form a concept $C4$ whose intent is {`m3`: `can_fly`}. According the expert, animals that can fly should also have wings (`m3` : `can_fly` $\prec$ `m4` : `has_wings`), `g5` : `dolphin` should not be grouped together with objects `g3`: `chicken` and `g6`: `honeybee` to form a concept. This is better represented by concept C_{12} whose extent is {`g3`: `chicken`, `g6`: `honeybee`} and intent is {`m2`: `lays_eggs`, `m3`: `can_fly`, `m4`: `has_wings`}. By checking the change C_4 in $\mathcal{L}$ to C_{12} in $\mathcal{L}_1$, we found that the data contain a noisy element: dolphins can fly.

Projections for Constrained Lattices w.r.t. Dependencies between Attribute Sets

To deal with dependencies between attribute sets, we provide a generalization of the definition of attribute dependencies introduced in [Belohlavek and Sklenar, 2005]. This definition can be applied to both single attributes and attribute sets.

Definition 4.6 (Dependency between Attribute Sets)**.** A *dependency* between attribute sets X and Y is in the form $X \prec Y$, where attribute set X is less important than attribute set Y, and the presence of X is not meaningful without the presence of Y.

Definition 4.7 (Formal Concept Satisfaction)**.** A formal concept (A, B) satisfies a dependency $X \prec Y$ between attribute sets X and Y iff whenever $X \subseteq B$ then $Y \subseteq B$.

Example 4.8. Consider the concept lattice as shown in Figure 4.2. In this lattice, concept C_{18} whose intent is {`m1, m2, m3, m4, m6, m8`} satisfies the dependency {`m1, m2`} $\prec$ {`m3`} because its intent contains both the attribute sets {`m1, m2`} and {`m3`}. Instead, concept C_{14} whose intent is {`m1, m2, m4, m6, m8`} and concept C_{20} whose intent is {`m1, m2, m4, m6, m8, m9`} do not satisfy the dependency {`m1, m2`} $\prec$ {`m3`} because their intent contains the attribute set {`m1, m2`}, but not the attribute set {`m3`}.

Definition 4.8 (Constrained Concept Lattice)**.**

(1) A concept lattice $\mathcal{L}$ constrained by a dependency $X \prec Y$ between attribute sets X and Y, is the collection of all formal concepts from lattice $\mathcal{L}$ which satisfy $X \prec Y$.

(2) A concept lattice $\mathcal{L}$ constrained by a set of dependencies D, is the collection of all formal concepts from lattice $\mathcal{L}$ which satisfy all the dependencies in D.

Example 4.9. Consider the concept lattice shown in Figure 4.2. The lattice constrained by the dependency {`m1, m2`} $\prec$ {`m3`} is the collection of all the formal concepts from the lattice shown in Figure 4.2 which satisfy this dependency, i.e. the collection of the formal concepts circled blue in Figure 4.8.

According to Definition 4.1 of attribute implications, if an implication $X \rightarrow Y$ between attribute sets X and Y holds in a lattice, then that lattice satisfies the dependency $X \prec Y$. Thus, similar to dependencies between single attributes, our mapping function in the case of dependencies between attribute sets is an extensional projection that gives a projected lattice satisfying an implication between these attribute sets.

Given a concept lattice $\mathcal{L}$ which does not satisfy the implication $X \rightarrow Y$ between attribute sets $X, Y \subseteq M$, the extensional projection for generating the projected lattice that satisfies the implication $X \rightarrow Y$ from $\mathcal{L}$ is:

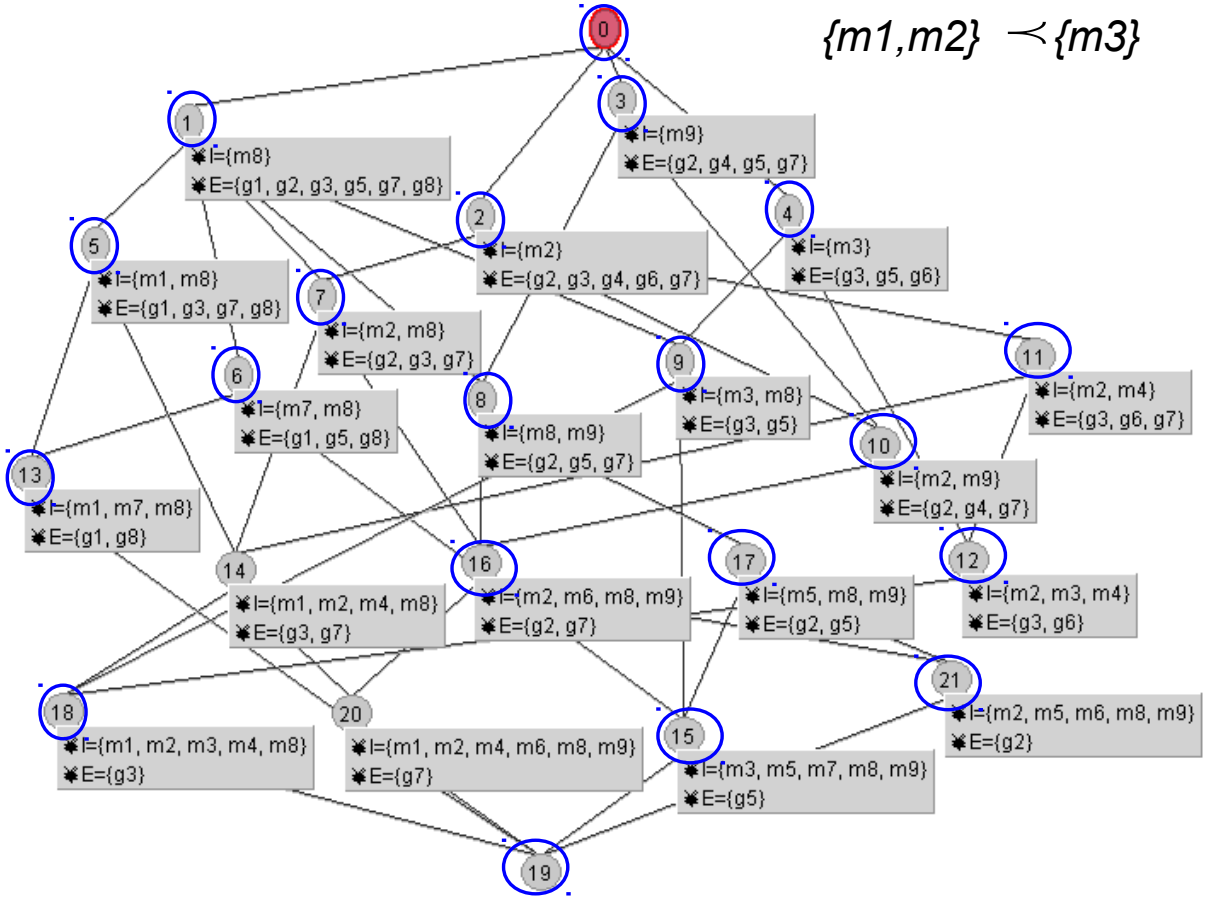

Figure 4.8: The lattice constrained by {`m1, m2`} ≺ {`m3`}.

$$\psi(A) = \begin{cases} A \cap (X' \cap Y') & \text{if } A \subseteq X' \text{AND } A \not\subseteq (X' \cap Y'), \\ A & \text{otherwise.} \end{cases} \tag{4.4}$$

Because the properties of attribute implications (Propositions 4.1 and 4.2) and projections in lattices are true for both single attributes and attribute sets, the properties of the projection in case of single attributes can be extended to attribute sets. The following propositions state the properties of the projection ψ in the case of attribute sets.

Proposition 4.8. Let $\mathcal{L}$ be a concept lattice which does not satisfy the implication $X \to Y$ between sets of attributes X and Y, then:
$X' \not\subseteq Y' \Longrightarrow X' \cap Y' \subset X'$

Proposition 4.9. Let $\mathcal{L}$ be a concept lattice which does not satisfy the implication $X \to Y$ between attribute sets X and Y, and ψ be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies $X \to Y$, then:

1) $\psi(X')$ is the maximal set of objects having the attribute set X in the projected lattice,

2) $\psi(Y')$ is the maximal set of objects having the attribute set Y in the projected lattice, and

3) $\psi(X') \subseteq \psi(Y')$.

Proposition 4.10. Let $\mathcal{L}$ be a concept lattice which does not satisfy the implication $X \to Y$ between attribute sets X and Y, and ψ be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies $X \to Y$, then:
$\psi(X') = X' \cap Y'$

Propositions 4.8, 4.9, 4.10 are proved in a similar way to Propositions 4.5, 4.6, 4.7 in the case of single attributes.

Given a concept lattice $\mathcal{L}$ which does not satisfy the implication $X \to Y$ between attribute sets X and Y, and an extensional projection ψ of lattice $\mathcal{L}$ such that $\psi(X') = X' \cap Y'$, extents A in lattice $\mathcal{L}$ can be divided into three categories shown in Figure 4.9.

- Category I contains all extents A that are subsumed by $X' \cap Y'$, i.e. $A \subseteq (X' \cap Y') \subset X'$.
- Category II contains all extents A that are subsumed by X' but not subsumed by $X' \cap Y'$, i.e. $A \subseteq X'$, $A \not\subseteq (X' \cap Y')$.
- Category III contains extents A that are not in parts I, II, i.e. $A \not\subseteq X'$.

Similar to the case of single attributes, by checking the behavior of the projection ψ according to its main properties on the parts I, II, III, we get the projection for generating the constrained lattice in Equation 4.4.

Example 4.10. Consider the running example where the expert has provided the dependency between attribute sets $\{\texttt{m1, m2}\} \prec \{\texttt{m3}\}$. According to the data in Table 4.1:

- $\{\texttt{m1, m2}\}' = \{\texttt{g3, g7}\}$,
- $\{\texttt{m3}\}' = \{\texttt{g3, g5, g6}\}$,
- $\{\texttt{m1, m2}\}' \cap \{\texttt{m3}\}' = \{\texttt{g3}\}$.

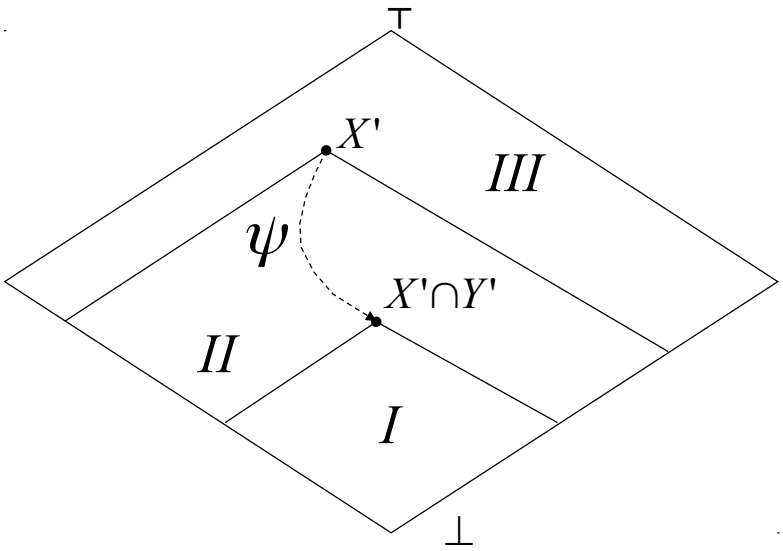

Figure 4.9: Three possible parts of extents in lattice $\mathcal{L}$ when $\psi(X') = X' \cap Y'$.

Applying Equation 4.4, we have the extensional projection ψ for generating the lattice constrained by the dependency $\{\texttt{m1, m2}\} \prec \{\texttt{m3}\}$ from the original lattice:

$$\psi(A) = \begin{cases} A \cap \{\texttt{g3}\} & \text{if } A \subseteq \{\texttt{g3, g7}\}, A \not\subseteq \{\texttt{g3}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.5}$$

Figure 4.10 depicts the constrained lattice and the trace of changes provided by the projection ψ. The formal concepts of the constrained lattice are circled blue. Applying Proposition 4.4, the trace of changes in this example contains C_{14} in $\mathcal{L}$ is changed to C_{18} in $\mathcal{L}_1$ and C_{20} in $\mathcal{L}$ is changed to C_{19} in $\mathcal{L}_1$.

An interpretation of the change C_{14} in $\mathcal{L}$ is changed to C_{18} in $\mathcal{L}_1$ according to the semantics of the projection is as follows. According to the data, object `g3`: `chicken` is grouped together with object `g7`: `penguin` to form a concept C_{14} whose intent is {`m1, m2, m4, m6, m8`}. According to the expert, $\{\texttt{m1, m2}\} \prec \{\texttt{m3}\}$. Instead, object `g3`: `chicken` should not be grouped together with object `g7`: `penguin` to form a concept. And thus, this is better represented by concept C_{18} whose extent is {`g3`: `chicken`} and intent is {`m1, m2, m3, m4, m6, m8`}. It is similar for the interpretation of the change C_{20} in $\mathcal{L}$ is changed to C_{19} in $\mathcal{L}_1$.

4.3.3 Projections for Constrained Lattices w.r.t. Sets of Dependencies

We consider the problem of generating constrained lattices and providing the trace of changes when experts provide their knowledge in the form of a set of dependencies.

A "naive" way to generate a constrained lattice that satisfies a set of implications consists in generating first a constrained lattice for each implication, and then getting the final constrained lattice by the intersection of these constrained lattices. This can be computed through the following proposition.

Proposition 4.11. Let $\mathcal{L}$ be a concept lattice, and ψ_i be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies an implication $X_i \to Y_i$ between attribute sets X_i and Y_i, then

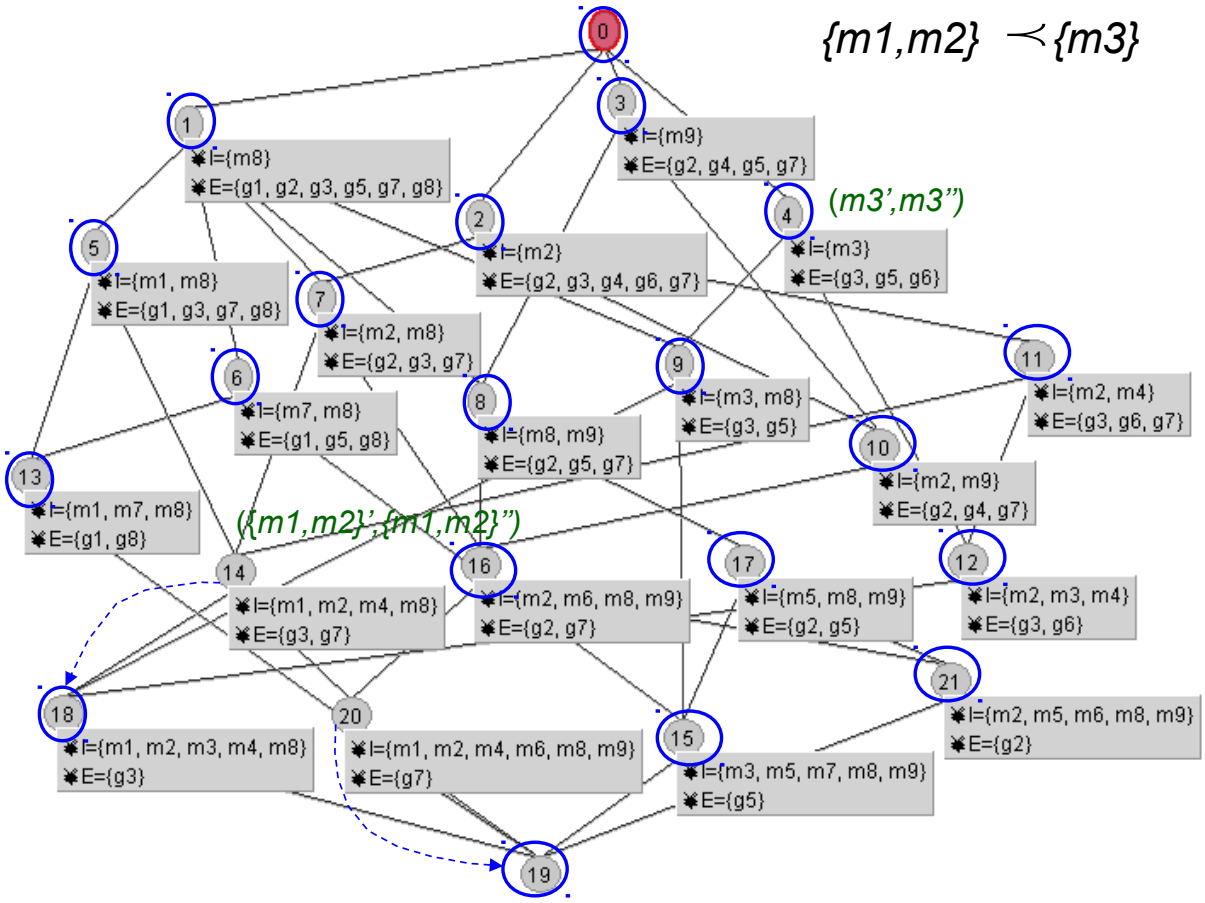

Figure 4.10: The lattice constrained by $\{\texttt{m1}, \texttt{m2}\} \prec \{\texttt{m3}\}$ and the trace of changes.

the constrained lattice that satisfies a set of implications $X_i \rightarrow Y_i$, where $i \in \{1, 2, ..., n\}$, is given by $\bigcap_{i=1}^{n} \psi_i(\mathcal{L})$.

A question raised here: is there a "good" way to generate a constrained lattice satisfying a set of dependencies? Given two dependencies, does it matter which one we apply first? In our situation, we have one original lattice and a set of projections. Can we have a "good" order to execute the set of projections? In the following, we will show that dependencies should be treated following an order of projections as introduced in [Soldano and Ventos, 2011].

Definition 4.9 ([Soldano and Ventos, 2011])**.** Let $\mathcal{L}$ be a concept lattice and ψ_1, ψ_2 be two projections of $\mathcal{L}$, we say that $\psi_1 \leq \psi_2$, iff if there is some projection ψ defined on $\psi_2(\mathcal{L})$ such that for all A in $\mathcal{L}$, $\psi_1(A) = \psi \circ \psi_2(A)$.

Definition 4.9 states that actually projections can be ordered from less "general" to more "specific". If ψ_1 is a projection over the projected lattice of ψ_2, then we say ψ_1 is more *specific* than ψ_2 or ψ_2 is more *general* than ψ_1. Indeed, if ψ_1 is a projection over the projected lattice of ψ_2, then ψ_1 is more specific than ψ_2. This is explained more detail in the following proposition introduced in [Buzmakov et al., 2015].

Proposition 4.12 ([Buzmakov et al., 2015])**.** Given a concept lattice $\mathcal{L}$, and ψ_1, ψ_2 be two projections of $\mathcal{L}$, if $\psi_1(\mathcal{L}) \subseteq \psi_2(\mathcal{L})$, then $\psi_1 \leq \psi_2$.

There is a partial order on projections of a lattice given by Proposition 4.13. This proposition has been proven in [Buzmakov et al., 2015].

Proposition 4.13 ([Buzmakov et al., 2015])**.** Projections of a lattice $\mathcal{L}$ ordered by Definition 4.9 form a semi-lattice $(\mathcal{F}, \wedge)$, where the semi-lattice operation between $\psi_1, \psi_2 \in \mathcal{F}$ is given by $\psi_1 \wedge \psi_2 = \psi_3$ iff $\psi_3(\mathcal{L}) = \psi_1(\mathcal{L}) \cap \psi_2(\mathcal{L})$.

The order on projections for generating constrained lattices can be characterized by Propositions 4.14.

Proposition 4.14. Let $\mathcal{L}$ be a concept lattice, ψ_1 be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies the implication $X_1 \rightarrow Y_1$, and ψ_2 be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies the implication $X_2 \rightarrow Y_2$, where $X_1, Y_1, X_2, Y_2 \subseteq M$, we have:

1) If $X_1 \subseteq X_2$ and $Y_1 \subseteq Y_2$, then $\psi_1 \leq \psi_2$.

2) If $\psi_1 \leq \psi_2$, then the projected lattice given by ψ_1 satisfies $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$.

3) There exists an extensional projection ψ_3 of $\mathcal{L}$ such that $\psi_3(\mathcal{L}) = \psi_1(\mathcal{L}) \cap \psi_2(\mathcal{L})$. ψ_3 gives the constrained lattice that satisfies $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$.

Proof.

1) Let $\psi_1(\mathcal{L})$ be the projected lattice given by ψ_1, $\psi_2(\mathcal{L})$ be the projected lattice given by ψ_2, because $\psi_1 \leq \psi_2$, according to Definition 4.9 of orders between projections, there is some projection ψ defined on $\psi_2(\mathcal{L})$ such that for all A in $\mathcal{L}$, $\psi_1(A) = \psi \circ \psi_2(A)$. Hence, $\psi_1(\mathcal{L}) \subseteq \psi_2(\mathcal{L})$. So, the projected lattice given by ψ_1 satisfies $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$.

2) In order to give proof that $\psi_1 \leq \psi_2$, we will show that a projection ψ can be defined on $\psi_2(\mathcal{L})$ such that $\psi_1(\mathcal{L}) = \psi \circ \psi_2(\mathcal{L})$:

$$\psi_1(A) = \begin{cases} \psi_2(A) \cap (X_1' \cap X_2') & \text{if } A \not\subseteq X_2', A \not\subseteq X_1' \cap Y_1', A \subseteq X_1', \\ \psi_2(A) & \text{otherwise.} \end{cases} \tag{4.6}$$

3) This follows from Proposition 4.13 that the set of projections $\mathcal{F}$ over $\mathcal{L}$ is a semi-lattice, then the meet $\psi_3 = \psi_1 \wedge \psi_2$ must exist.

□

As a result of Proposition 4.14, given two dependencies between attribute sets, we can order the projections corresponding to these dependencies as follows. First, if one dependency depends on attribute sets that are included in the attribute sets of the other dependency, then the projection of that dependency is more specific than the projection of the other. Second, if one projection is more specific than the other, then we can use the more specific projection for generating the final constrained lattice instead of using all the projections. Third, we can use the projection that is the meet of the two projections for generating the final constrained lattice instead of using all the projections.

Let us now go back to our scenario of generating a constrained lattice that satisfies a set of dependencies. Let $\mathcal{L}$ be a concept lattice, and ψ_i be an extensional projection of $\mathcal{L}$ such that the projected lattice satisfies an implication $X_i \to Y_i$, where $X_i, Y_i \subseteq M$. The set of projections ψ_i can be ordered according to the order of projections given by Propositions 4.14. This order forms a semi-lattice given by Propositions 4.13. By Propositions 4.14, the projection that is the meet of the most specific projections in this order gives the final constrained lattice satisfying the set of implications.

According to the order of projections, we have two possible ways of generating constrained lattices. The first way uses the meet of the most specific projections in the order of the set of projections to generate the final constrained lattice. The second way uses all the projections to generate a set of constrained lattices. Applying the first or the second way to generate constrained lattices depends on what experts need. The first way using the meet of the most specific projections to generate the final constrained lattice is more efficient in computation than the second way using all the projections to generate the corresponding constrained lattices. However, by using the meet of the most specific projections to generate the final constrained lattice, the first way only provides the trace of changes between the original lattice and the final constrained lattice. In the case experts need all the traces of changes, we need the second way using all the projections to generate the corresponding constrained lattices.

Example 4.11. Consider the running example where the expert provides her knowledge in the form of a set of dependencies d_i:

d_1) {m3, m8} $\prec$ {m4},

d_2) m3 $\prec$ m4,

d_3) {m1, m2} $\prec$ {m3},

d_4) m4 $\prec$ m3.

Let ψ_i be the extensional projection for generating the constrained lattice satisfying dependency d_i, applying Equation 4.4, we have:

$$\texttt{For } d_1 : \psi_1(A) = \begin{cases} A \cap \{\texttt{g3}\} & \text{if } A \subseteq \{\texttt{g3, g5}\}, A \not\subseteq \{\texttt{g3}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.7}$$

$$\texttt{For } d_2 : \psi_2(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g5, g6}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.8}$$

$$\texttt{For } d_3 : \psi_3(A) = \begin{cases} A \cap \{\texttt{g3}\} & \text{if } A \subseteq \{\texttt{g3, g7}\}, A \not\subseteq \{\texttt{g3}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.9}$$

$$\texttt{For } d_4 : \psi_4(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g6, g7}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.10}$$

Lattices *i*, *ii*, *iii*, *iv* in Figure 4.14 depict the constrained lattices and the traces of changes provided by the projections ψ_1, ψ_2, ψ_3, ψ_4 respectively. The details of these constrained lattices are shown in Figures 4.11, 4.7, 4.10, 4.6 respectively. In this set of projections, $\psi_2 \leq \psi_1$ and $\psi_4 \leq \psi_3$. We can see that the constrained lattice $\psi_2(\mathcal{L})$ is included in $\psi_1(\mathcal{L})$ and $\psi_4(\mathcal{L})$ is included in $\psi_3(\mathcal{L})$.

Let ψ_5 be an extensional projection that is the meet of the most specific projections: $\psi_5 = \psi_2 \wedge \psi_4$, ψ_5 can be defined such that:

$$\psi_5(A) = \begin{cases} A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g6, g7}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A \cap \{\texttt{g3, g6}\} & \text{if } A \subseteq \{\texttt{g3, g5, g6}\}, A \not\subseteq \{\texttt{g3, g6}\}, \\ A & \text{otherwise.} \end{cases} \tag{4.11}$$

The set of projections ψ_i forms a semi-lattice as shown in Figure 4.12. We have two ways of generating the final constrained lattice. The first way uses the meet ψ_5 of the most specific projections. Lattice *v* in Figure 4.14 depicts the final constrained lattice and the trace of changes between the original lattice and the final constrained lattice provided by the projection ψ_5. Figure 4.13 depicts the details of the final constrained lattice and the trace of changes. The second way uses all the projections to have all the traces of changes. According to the semi-lattice of the projections, because $\psi_2 \leq \psi_1$ and $\psi_4 \leq \psi_3$, in order to get all the traces of changes, ψ_1 is applied before ψ_2 and ψ_3 is applied before ψ_4. By contrast, as ψ_2 and ψ_4 are incompatible, it does not matter if ψ_1 and ψ_2 or ψ_3 and ψ_4 are applied first. Thus, the projections can be applied according to either the order ψ_1,ψ_2,ψ_3,ψ_4,ψ_5 or ψ_3,ψ_4.ψ_1,ψ_2,ψ_5. The trace of changes can be either the chain of lattices *i*, *ii*, *iii*, *iv*, *v* as shown in Figure 4.14 or in Figure 4.15. In both cases, experts still can access the changes corresponding to each dependency.

4.3.4 The Framework for Generating Constrained Lattices

We now have the formal framework for generating a lattice that is constrained by experts' knowledge in the form of dependence relations between attributes and providing the trace of changes. This framework is based on the extensional projections that we have presented above.

The framework supports domain experts to provide their knowledge in the form of dependencies between attribute sets according to Definition 4.6 and offers to them two ways of generating constrained lattices that depend on how they want to get the trace of changes. The first task in this framework consists of converting each dependency into an extensional projection given

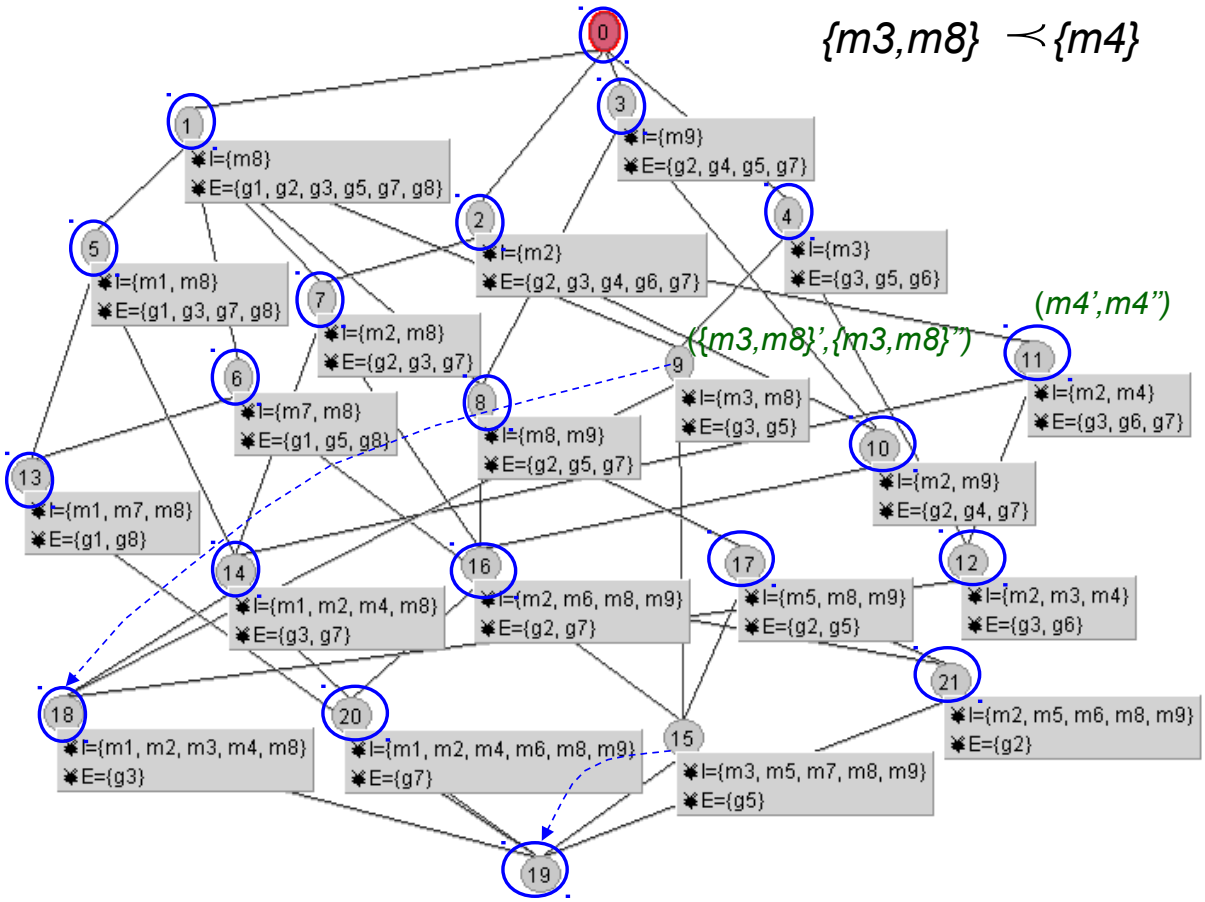

Figure 4.11: The lattice constrained by {m3, m8} ≺ {m4} and the trace of changes.

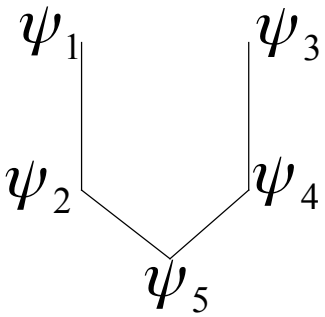

Figure 4.12: The semi-lattice of the projections.

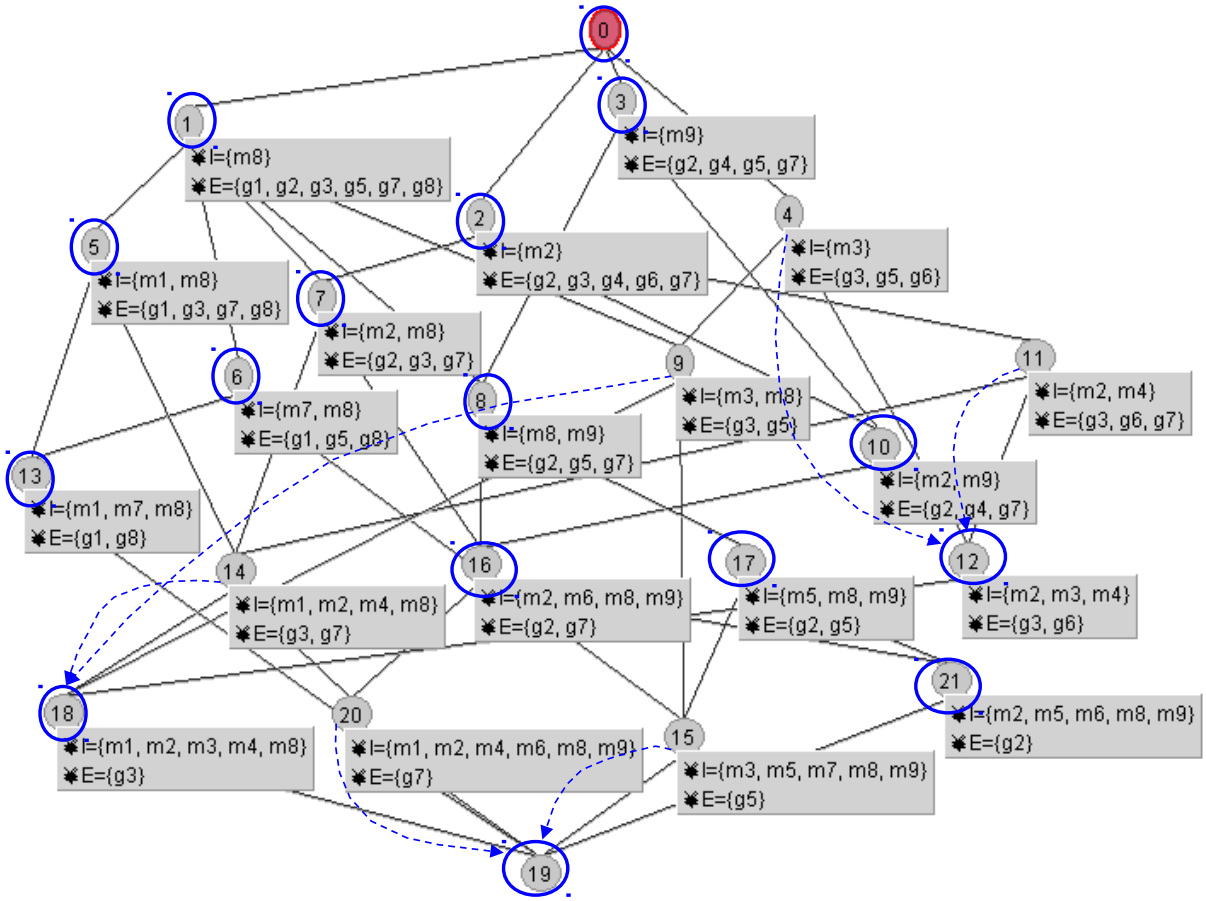

Figure 4.13: The final constrained lattice and the trace of changes.

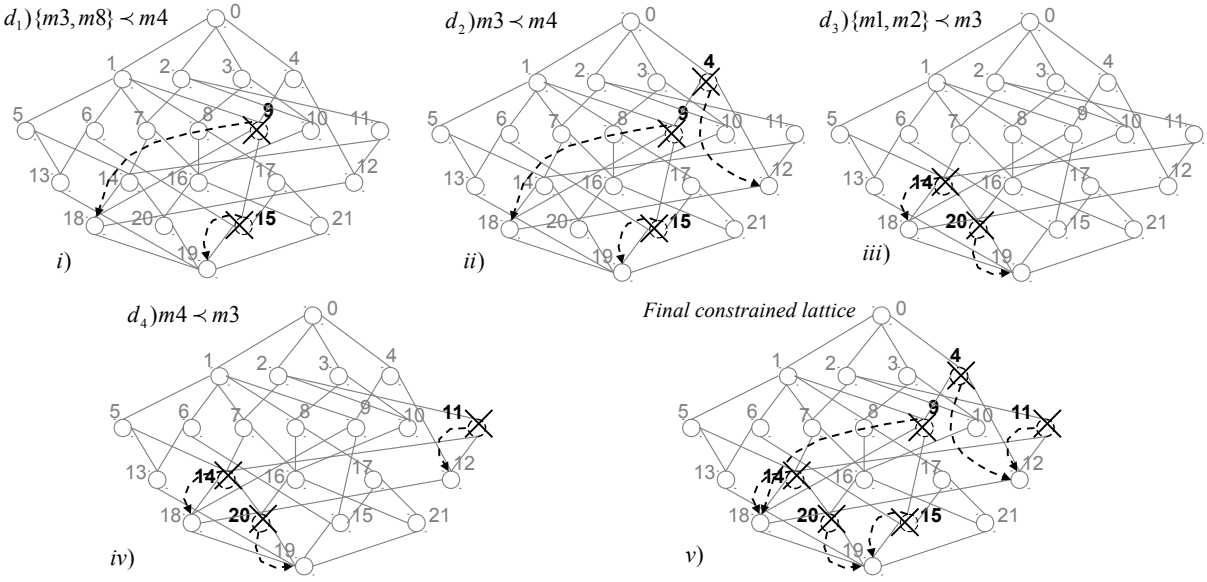

Figure 4.14: The final constrained lattice and the trace of changes.

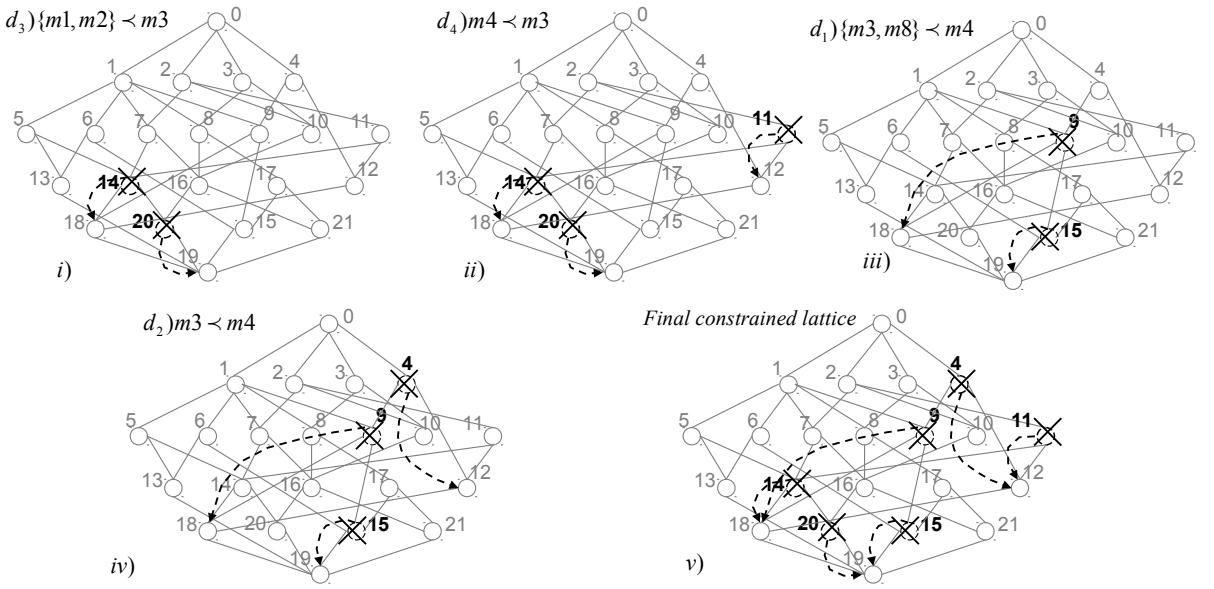

Figure 4.15: The final constrained lattice and the trace of changes.

by Equation 4.4. In the situation such that, experts do not care about the trace of changes corresponding to each dependency, the set of projections is composed according to the order of projections given by Propositions 4.14. Then, the meet of the most specific projections in the order of the set of projections is used for generating the constrained lattice and providing the trace of changes. In this situation, the computation for generating the constrained lattice is more efficient, but experts can see only the trace of changes between the original lattice and the final constrained lattice. In the other situation, experts care about the trace of changes corresponding to each dependency, all the projections corresponding to the dependencies are used for generating the corresponding constrained lattices and providing the trace of changes. In this situation, the computation for generating the constrained lattice is less efficient, but experts can have all the traces of changes.

4.4 Related Work, Discussion and Conclusion

Taking into account expert knowledge in the form of the dependencies between attributes in concept lattices has been proposed by several researchers [Ganter, 1987, Wille, 1989, Stumme, 1996, Ganter, 1999, Carpineto and Romano, 2004, Belohlavek and Sklenar, 2005]. *Attribute exploration* [Ganter, 1987, Wille, 1989] is a well-known knowledge acquisition method that allows computing the implication base interactively. The attribute exploration procedure calculates a minimal set of implications by asking users to provide objects that are counterexamples for the invalid implications in the context. Later, some researchers [Stumme, 1996, Ganter, 1999] extended attribute exploration to include background implications, i.e. the implications that experts already know to be valid. In these approaches, they trust the original data and experts have to provide new objects as counterexamples. To deal with a situation such that experts know dependencies between attributes, but they do not know any new objects to provide, the other approaches [Carpineto and Romano, 2004, Belohlavek and Sklenar, 2005] were proposed to build a concept hierarchy from a formal context extracted from data and from a hierarchy of attributes provided by domain experts. Carpineto and Romano [Carpineto and Romano, 2004] discussed

two ways for adding knowledge to object descriptions. One possible way is to expand the formal context by adding to the description of each object all attributes that are implied by the original attributes. Expanding the context can lead to lose the original data and the formal context may become very large. The other way is to work with an unexpanded formal context by adapting the construction algorithms of lattices to extract formal concepts satisfying dependencies. A similar idea was proposed by Belohlavek and Skenar [Belohlavek and Sklenar, 2005]. In [Belohlavek and Sklenar, 2005], the authors introduced the notion of attribute dependencies for single attributes and proposed an algorithm adapted from the incremental algorithm AddIntent [Merwe et al., 2004] for computing constrained posets. By contrast, we do not expand the formal context nor adapt the construction algorithms of lattices. In our approach, we use projections to generate constrained lattices instead of constrained posets and to provide the trace of changes.

To conclude, in this chapter, we have presented a formal framework based on extensional projections for integrating expert knowledge into concept lattices in such a way that the lattice structure and the trace of changes occurring in the original lattice and the final constrained version are preserved. The expert knowledge is encoded as a set of attribute dependencies which is aligned with the set of implications provided by the concept lattice. The novel use of projections and the formal properties of attribute implications allow us to build the constrained lattices and provide the trace of changes, instead of providing only concepts that satisfy experts' knowledge. Thanks to the trace of changes, experts can access how concepts in practice are related to concepts automatically issued from data. According to the order of projections, the framework offers two ways of generating constrained lattices. The first way uses the meet of the most specific projections to generate the final constrained lattice. The second way uses all the projections to generate a set of constrained lattices. The first way is more efficient in computation, but provides only the trace of changes between the original lattice and the final constrained lattice while the second way provides all the traces of changes, but less efficient in computation.

The contributions of the work presented in this chapter are the following:

- A formal framework for generating lattices that are constrained by expert knowledge and providing the trace of changes. The result of constraining a lattice remains the lattice structure.
- A formalization of experts' knowledge for integrating into lattices using dependencies between attribute sets. This formalization can be applied to both single attributes and attribute sets. And thus, experts can have more possibilities in expressing constraints.
- The use of projections over lattices for a decoupling of data and expert knowledge, i.e. without changing data, from an original dataset, two different projections produce two different constrained lattices, and thus, the gap between the representation model based on a concept lattice and the representation model of a domain expert is filled with projections.
- An examination on the order of projections for generating constrained lattices w.r.t. a set of dependencies. Accordingly, we have two ways of using projections for generating constrained lattices. The first way uses the meet of the most specific projections in the order of the set of projections to generate the final constrained lattice. The second way uses all the projections to generate a set of constrained lattices. The first way is more efficient in computation, but provides only the trace of changes between the original lattice and the final constrained lattice while the second way provides all the traces of changes, but less efficient in computation.

Several interesting perspectives are opened following this work. Future work includes defining intensional projections in a similar way to integrate dependencies between sets of objects. It allows one to integrate a taxonomy of objects into concept lattices. This is useful for many applications such as classifying documents, this can be applied to integrate a taxonomy of documents from experts into concept lattices. Another interesting application would be to complete definitions in data. When we have two implications $X \to Y$ and $Y \to X$, we say that $X \iff Y$ is an equivalence or a definition [Alam et al., 2015]. When some implications are missing in data and experts know they are definitions, the approach using projections presented in this thesis can be applied to add implications to complete the data.

Appendix

$\psi(A) = A \cap (x' \cap y')$ complies with the properties of projections:

- monotone:
  $\forall A_1, A_2 \in$ Part II, $A_1 \subseteq A_2$:
  As $\psi(A_1) = A_1 \cap (x' \cap y')$ and $\psi(A_2) = A_2 \cap (x' \cap y')$, we have $\psi(A_1) \subseteq \psi(A_2)$.
- contractive:
  $\psi(A) = A \cap (x' \cap y') \subseteq A$.
- idempotent:
  As $\psi(A) = A \cap (x' \cap y')$, $\psi(\psi(A)) = \psi(A \cap (x' \cap y'))$.
  Because $A \cap (x' \cap y')$ belongs to Part I, $\psi(A \cap (x' \cap y')) = A \cap (x' \cap y') = \psi(A)$.
  So, $\psi(\psi(A)) = \psi(A)$.

Chapter 5

Conclusion and Perspectives

Contents

5.1 Summary

Formal Concept Analysis is a formal conceptualization method which has proved to be very efficient as a bottom-up approach for building ontologies. FCA elicits from data a class schema in the form of either a concept lattice or a set of attributes implications. The concept lattice can be interpreted as a knowledge model in the form of a concept hierarchy and the logical structures of formal concepts and concept lattices are effective in supporting human reasoning. However, building knowledge bases is a cognitive process and does not obey to strict and formal rules, domain experts may understand the domain in a different way than what is represented in data. Thus, often there exists a gap between the representation model based on a concept lattice and the representation model of a domain expert.

In this thesis, we have presented our contributions to knowledge extraction and semantic annotation using Formal Concept Analysis. In this work, we focus on helping domain experts in building the knowledge model based on a concept lattice from semantic annotations of textual resources. Our work also aims at improving the annotation process: annotations of texts can be modified or enhanced to make the resulting knowledge model as close as possible to the requirements of domain experts.

Our first contribution is based on a clever methodology design for interactive and iterative extracting knowledge from textual resources that unifies knowledge extraction and semantic annotation into one single process. Semantic annotations are used for formalizing the source of knowledge in texts and keeping the traceability between the knowledge model and the source of knowledge. The methodology, named KESAM (Knowledge Extraction and Semantic Annotation Management), uses Formal Concept Analysis as a core engine for building the knowledge model, i.e. the concept lattice, iteratively and ensuring the link between semantic annotations and each knowledge units in the concept lattice. At the first stage, we have shown how the KESAM system can be used to build the lattice from semantic annotations of texts. In order to get the resulting lattice as close as possible to the requirements of domain experts, the KESAM process enables expert interaction on the formal context, the concept lattice and semantic annotations. To enable

expert interaction in the system, we introduce sets of changes in the lattice, the formal context, and semantic annotations. Domain experts can evaluate and make changes in the lattice, the formal context, or semantic annotations until they reach an agreement between the knowledge model and their own knowledge or requirements. The changes are implemented following the incremental lattice building approaches for managing the consequences of the changes in the lattice. When domain experts ask for a change, its consequence in the lattice is reported to domain experts so that domain experts can decide if that change should be applied. The KESAM process then is in charge of keeping the formal context, the concept lattice and semantic annotations up to date with the changes. The KESAM system, therefore keeps the consistency between the knowledge model based on the concept lattice and semantic annotations, and converges towards a knowledge model close to the requirements of domain experts. Thanks to the link between the knowledge model and semantic annotations, the traceability between the knowledge model and semantic annotations can be enabled; semantic annotations and the knowledge model can be efficiently updated. Moreover, FCA allows to build the definitions of concepts with a sets of objects and sets of attributes and therefore, knowledge extraction and semantic annotation in the KESAM system can work with both atomic and defined concepts.

Our second contribution is a formal method for bridging the possible gap between the representation model based on a concept lattice and the representation model of a domain expert. The knowledge of the expert is encoded as a set of attribute dependencies or a set of constraints which are "aligned" with the set of implications provided by the concept lattice leading to its later modification. The method can be generalized for generating lattices guided by constraints based on attribute dependencies and using extensional projections. This method also allows the experts to keep a trace of the changes occurring in the original lattice and the final constrained version, and to assess how concepts in practice are related to concepts automatically issued from data. For generating constrained lattices w.r.t. a set of dependencies, according to the order of projections, the method offers two possible ways of generating constrained lattices. The first way uses the meet of the most specific projections to generate the final constrained lattice. The second way uses all the projections to generate a set of constrained lattices. The first way is more efficient in computation, but provides only the trace of changes between the original lattice and the final constrained lattice while the second way provides all the traces of changes, but less efficient in computation.

5.2 Perspectives

In Chapter 3, we have presented KESAM, a methodology for interactive and iterative extracting knowledge from textual resources using Formal Concept Analysis. The KESAM methodology benefits both knowledge extraction and semantic annotation. This work opens several perspectives. We can do more to help the domain experts. Refining the cost function associated with the changes can make easier the choice of change strategies. Tagging concepts that domain experts agree with and want to keep unchanged all along the process could reduce the number of the suggested strategies. Performing several changes at once also needs more investigations. Analyzing the lattice according to the metrics for concept evaluation in ontologies [Alani et al., 2006, Peroni et al., 2008, Zhang et al., 2010, Tartir et al., 2010] and lattices [Kuznetsov and Makhalova, 2015] such as *frequency* [Stumme, 2002], *stability* [Kuznetsov, 2007, Fennouh et al., 2014], *weights* [Belohlavek and Macko, 2011, Belohlavek and Trnecka, 2013], *ranking concept by matrix factorization* [Piskova et al., 2014], etc. would be meaningful for guiding experts in the evaluation process, especially in identifying the concepts that should be refined. Besides, we

should provide experts with an environment for searching new texts with respect to their own knowledge and introducing new annotations automatically or manually.

In Chapter 4, we have presented a formal method for bridging the possible gap between the representation model based on a concept lattice and the representation model of a domain expert using extensional projections. The method allows to integrate a partial order of attributes from domain experts into concept lattices and provides a trace of the changes occurring in the original lattice and the revised version. Several interesting perspectives are opened following this work. An interesting perspective of this work is to define intensional projections in a similar way to integrate dependencies between sets of objects in concept lattices. It allows one to integrate a partial order of objects into concept lattices. This is useful for many applications such as classifying documents, this can be applied to integrate a partial order of documents from experts into concept lattices. Moreover, it would be interesting to see how the domain knowledge in a complex form of dependencies such as a disjunction, e.g. *expert systems imply computer applications or knowledge-based systems*, can be integrated in the concept lattice. Another interesting application would be to complete definitions in data. When we have two implications $X \rightarrow Y$ and $Y \rightarrow X$, we say that $X \Longleftrightarrow Y$ is an equivalence or a definition [Alam et al., 2015]. The approach using projections presented in this thesis can be applied to add implications that domain experts expect to exist to complete the data. In addition, predicting the association rules in data can be "real" implications for recommending domain experts is a practical perspective.

Bibliography

[Alam et al., 2015] Alam, M., Buzmakov, A., Codocedo, V., and Napoli, A. (2015). Mining definitions from RDF annotations using formal concept analysis. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31*, pages 823–829.

[Alani et al., 2006] Alani, H., Brewster, C., and Shadbolt, N. (2006). Ranking ontologies with aktiverank. In Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., and Aroyo, L., editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg.

[Appelt et al., 1995] Appelt, D., Hobbs, J., Bear, J., Israel, D., Kameyama, M., Kehler, A., Martin, D., Myers, K., and Tyson, M. (1995). SRI International FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 237–248, San Francisco. Morgan Kaufmann.

[Arnold et al., 2008] Arnold, A., Nallapati, R., and Cohen, W. (2008). Exploiting feature hierarchy for transfer learning in named entity recognition. In McKeown, K., Moore, J., Teufel, S., Allan, J., and Furui, S., editors, *ACL*, pages 245–253. The Association for Computer Linguistics.

[Aussenac-Gilles et al., 2008] Aussenac-Gilles, N., Després, S., and Szulman, S. (2008). The terminae method and platform for ontology engineering from texts. In Buitelaar, P. and Cimiano, P., editors, *Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text*, pages 199–223. IOS Press.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA.

[Barbut and Monjardet, 1970] Barbut, M. and Monjardet, B. (1970). *Ordre et classification, Algèbre et combinatoire*. Hachette, Paris.

[Bayerl et al., 2003] Bayerl, P., Lüngen, H., Gut, U., and Paul, K. (2003). Methodology for reliable schema development and evaluation of manual annotations. In *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (K-CAP 2003*, pages 17–23.

[Belohlavek and Macko, 2011] Belohlavek, R. and Macko, J. (2011). Selecting important concepts using weights. In *Proceedings of the 9th International Conference on Formal Concept Analysis (ICFCA)*, pages 65–80, Berlin, Heidelberg. Springer-Verlag.

[Belohlavek and Sklenar, 2005] Belohlavek, R. and Sklenar, V. (2005). Formal concept analysis constrained by attribute-dependency formulas. In Ganter, B. and Godin, R., editors, *Formal Concept Analysis*, volume 3403 of *Lecture Notes in Computer Science*, pages 176–191. Springer Berlin Heidelberg.

[Belohlavek and Trnecka, 2013] Belohlavek, R. and Trnecka, M. (2013). Basic level in formal concept analysis: Interesting concepts and psychological ramifications. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1233–1239. AAAI Press.

[Bendaoud et al., 2008a] Bendaoud, R., Napoli, A., and Toussaint, Y. (2008a). Formal concept analysis: A unified framework for building and refining ontologies. In Gangemi, A. and Euzenat, J., editors, *Proceedings of International Conference on Knowledge Engineering and Knowledge Management(EKAW)*, volume 5268 of *Lecture Notes in Computer Science*, pages 156–171. Springer.

[Bendaoud et al., 2008b] Bendaoud, R., Napoli, A., and Toussaint, Y. (2008b). A proposal for an interactive ontology design process based on formal concept analysis. In Eschenbach, C. and Grüninger, M., editors, *5th international conference on Formal Ontology in Information Systems (FOIS)*, volume 183 of *Frontiers in Artificial Intelligence and Applications*, pages 311–323, Saarbrücken, Germany.

[Bendaoud et al., 2008c] Bendaoud, R., Toussaint, Y., and Napoli, A. (2008c). Pactole: A methodology and a system for semi-automatically enriching an ontology from a collection of texts. In *Proceedings of the 16th international conference on Conceptual Structures: Knowledge Visualization and Reasoning (ICCS)*, pages 203–216, Berlin, Heidelberg. Springer-Verlag.

[Blansché et al., 2010] Blansché, A., Skaf-Molli, H., Molli, P., and Napoli, A. (2010). Human-machine collaboration for enriching semantic wikis using formal concept analysis. In *5th Workshop on Semantic Wikis Linking Data and People - SemWiki2010*, Heraklion, Greece.

[Bontcheva and Cunningham, 2011] Bontcheva, K. and Cunningham, H. (2011). Semantic annotations and retrieval: Manual, semiautomatic, and automatic generation. In *Handbook of Semantic Web Technologies*, pages 77–116. Springer Berlin Heidelberg.

[Bordat, 1986] Bordat, J. (1986). Calcul pratique du treillis de galois d'une correspondance. *Informatiques et Sciences Humaines*, 96:31–47.

[Borst, 1997] Borst, W. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, Institute for Telematica and Information Technology, University of Twente, Enschede, The Netherlands.

[Bozsak et al., 2002] Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Stumme, G., Sure, Y., Tane, J., Volz, R., and Zacharias, V. (2002). KAON - towards a large scale Semantic Web. In Bauknecht, K., Tjoa, A., and Quirchmayr, G., editors, *Proc. E-Commerce and Web Technologies, Third International Conference*, number 2455 in Lecture Notes in Computer Science, Aix-en-Provence, France. Springer.

[Brachman and Anand, 1996] Brachman, R. and Anand, T. (1996). Advances in knowledge discovery and data mining. chapter The Process of Knowledge Discovery in Databases, pages 37–57. American Association for Artificial Intelligence, Menlo Park, CA, USA.

[Brewster et al., 2003] Brewster, C., Ciravegna, F., and Wilks, Y. (2003). Background and foreground knowledge in dynamic ontology construction viewing text as knowledge maintenance. In Dieng-Kuntz, R. and Gandon, F., editors, *KCAP 2003 Workshop on Knowledge Management and the Semantic Web*, pages 9–16. KCAP. A revised version of a paper presented at SIGIR 03.

[Buitelaar et al., 2009] Buitelaar, P., Cimiano, P., Haase, P., and Sintek, M. (2009). Towards linguistically grounded ontologies. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 111–125. Springer Berlin Heidelberg.

[Buitelaar et al., 2005] Buitelaar, P., Cimiano, P., and Magnini, B., editors (2005). *Ontology Learning from Text: Methods, Evaluation and Applications*. Amsterdam: IOS Press.

[Buzmakov et al., 2015] Buzmakov, A., Kuznetsov, S., and Napoli, A. (2015). Revisiting pattern structure projections. In Baixeries, J., Sacarea, C., and Ojeda-Aciego, M., editors, *Formal Concept Analysis*, volume 9113 of *Lecture Notes in Computer Science*, pages 200–215. Springer International Publishing.

[Carpineto and Romano, 2004] Carpineto, C. and Romano, G. (2004). Adding knowledge to concept lattices. In *Concept Data Analysis: Theory and Applications*, pages 64–81. John Wiley & Sons.

[Chandrasekaran et al., 1999] Chandrasekaran, B., Josephson, J., and Benjamins, V. (1999). What are ontologies, and why do we need them? *Intelligent Systems and their Applications, IEEE*, 14(1):20–26.

[Chein, 1969] Chein, M. (1969). Algorithme de recherche des sous-matrices premières d'une matrice. *Bull. Math. Soc. Sc. Math. de Roumanie*, 1(13):21–25.

[Cimiano, 2006] Cimiano, P. (2006). *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag New York, Inc.

[Cimiano et al., 2004a] Cimiano, P., Handschuh, S., and Staab, S. (2004a). Towards the self-annotating web. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 462–471, New York, NY, USA. ACM.

[Cimiano et al., 2004b] Cimiano, P., Hotho, A., and Staab, S. (2004b). Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text. In *Proceedings of the 16th European Conference on Artificial Intelligence*, Valencia, Spain.

[Cimiano et al., 2005] Cimiano, P., Hotho, A., and Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis. volume 24, pages 305–339, USA. AI Access Foundation.

[Cimiano et al., 2004c] Cimiano, P., Stumme, G., Hotho, A., and Tane, J. (2004c). Conceptual knowledge processing with formal concept analysis and ontologies. In *Proceedings of the The Second International Conference on Formal Concept Analysis (ICFCA)*, volume 2961 of *Lecture Notes in Artificial Intelligence*, pages 189–207, Sydney, Australia.

[Cimiano and Völker, 2005] Cimiano, P. and Völker, J. (2005). Text2onto - a framework for ontology learning and data-driven change discovery. In Springer, editor, *Proceedings of Natural*

Language Processing and Information Systems (NLDB), number 3513 in Lecture Notes in Computer Science, pages 227–238.

[Cimiano and Völker, 2005] Cimiano, P. and Völker, J. (2005). Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*, pages 166–172.

[Davies, 2010] Davies, J. (2010). Lightweight ontologies. In Poli, R., Healy, M., and Kameas, A., editors, *Theory and Applications of Ontology: Computer Applications*, pages 197–229. Springer Netherlands.

[Declerck and Lendvai, 2010] Declerck, T. and Lendvai, P. (2010). Towards a standardized linguistic annotation of the textual content of labels in knowledge representation systems. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 3836–3839, Valletta, Malta.

[Diday, 1982] Diday, E. (1982). *Éléments d'analyse de données*. Dunod décision. Dunod.

[Dingli et al., 2003] Dingli, A., Ciravegna, F., and Wilks, Y. (2003). Automatic semantic annotation using unsupervised information extraction and integration. In Handschuh, S., Koivunen, M.-R., Dieng, R., and Staab, S., editors, *Proceedings of the Workshop on Semantic Annotation and Knowledge Markup, SEMANNOT2003, K-CAP 2003 Workshop, at Sanibel, 26 October 2003*.

[Dunham, 2002] Dunham, M. (2002). *Data Mining: Introductory and Advanced Topics*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[Duquenne and Guigues, 1986] Duquenne, V. and Guigues, J. (1986). Familles minimales d'implications informatives résultant d'un tableau de données binaires. *Mathématiques et Sciences Humaines*, 95:5–18.

[Dzyuba et al., 2013] Dzyuba, V., van Leeuwen, M., Nijssen, S., and Raedt, L. D. (2013). Active preference learning for ranking patterns. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 532–539.

[Dzyuba et al., 2014] Dzyuba, V., van Leeuwen, M., Nijssen, S., and Raedt, L. D. (2014). Interactive learning of pattern rankings. *International Journal on Artificial Intelligence Tools*, 23(6).

[Ekbal and Bandyopadhyay, 2010] Ekbal, A. and Bandyopadhyay, S. (2010). Named entity recognition using support vector machine: a language independent approach. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 4:589 – 604.

[Euzenat, 2002] Euzenat, J. (2002). Eight questions about semantic web annotations. *IEEE Intelligent Systems*, 17:55–62.

[Fayyad et al., 1996] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54.

[Fennouh et al., 2014] Fennouh, S., Nkambou, R., Valtchev, P., and Rouane-Hacene, M. (2014). Stability-based filtering for ontology restructuring. In *Proceedings of the 12th International Conference on Formal Concept Analysis (ICFCA)*.

[Finkel and Manning, 2009] Finkel, J. and Manning, C. (2009). Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 1 of *EMNLP '09*, pages 141–150, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Furst and Trichet, 2006] Furst, F. and Trichet, F. (2006). Heavyweight ontology engineering. In *Proceedings of the International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE)*, Montpellier, France.

[Gaizauskas et al., 1995] Gaizauskas, R., Wakao, T., Humphreys, K., Cunningham, H., and Wilks, Y. (1995). University of Sheffield: Description of the LaSIE system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 207–220, San Francisco. Morgan Kaufmann.

[Ganter, 1984] Ganter, B. (1984). Two basic algorithms in concept analysis. FB4–Preprint 831, TH Darmstadt.

[Ganter, 1987] Ganter, B. (1987). Algorithmen zur Formalen Begriffsanalyse. In Ganter, B., Wille, R., and Wolff, K. E., editors, *Beitr äge zur Begriffsanalyse*, pages 241–254. B.I. Wissenschaftsverlag.

[Ganter, 1999] Ganter, B. (1999). Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215 – 233. ORDAL'96.

[Ganter and Kuznetsov, 2001] Ganter, B. and Kuznetsov, S. (2001). Pattern structures and their projections. In Delugach, H. and Stumme, G., editors, *Conceptual Structures: Broadening the Base*, volume 2120 of *Lecture Notes in Computer Science*, pages 129–142. Springer Berlin Heidelberg.

[Ganter and Wille, 1999] Ganter, B. and Wille, R. (1999). *Formal Concept Analysis, Mathematical Foundations*. Springer.

[Ghahramani, 2002] Ghahramani, Z. (2002). Hidden markov models. chapter An Introduction to Hidden Markov Models and Bayesian Networks, pages 9–42. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

[Girault, 2008] Girault, T. (2008). Concept lattice mining for unsupervised named entity annotation. In *Proceedings of the Sixth International Conference on Concept Lattices and Their Applications*, Olomouc, Czech Republic.

[Godin et al., 1995] Godin, R., Missaoui, R., and Alaoui, H. (1995). Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence*, 11(2):246–267.

[Grishman and Sundheim, 1996] Grishman, R. and Sundheim, B. (1996). Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics*, volume 1 of *COLING '96*, pages 466–471, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Gruber, 1993] Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

[Guarino et al., 2009] Guarino, N., Oberle, D., and Staab, S. (2009). What is an ontology? In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Springer Berlin Heidelberg.

[Guénoche and Mechelen, 1993] Guénoche, A. and Mechelen, I. V. (1993). Galois approach to the induction of concepts. In Mechelen, I. V., Hampton, J., Michalski, R., and Theuns, P., editors, *Categories and Concepts: Theoretical Views and Inductive Data Analysis*. Academic Press.

[Hacene et al., 2013] Hacene, M. R., Huchard, M., Napoli, A., and Valtchev, P. (2013). Relational concept analysis: mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, 67(1):81–108.

[Handschuh and Staab, 2002] Handschuh, S. and Staab, S. (2002). Authoring and annotation of web pages in cream. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 462–473, New York, NY, USA. ACM.

[Huchard et al., 2007] Huchard, M., Napoli, A., Rouane-Hacene, M., and Valtchev, P. (2007). Mining description logics concepts with relational concept analysis. In Brito, P., Bertrand, P., Cucumel, G., and Carvalho, F. D., editors, *Selected Contributions in Data Analysis and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 259–270. Springer, Berlin.

[Humphreys et al., 1998] Humphreys, K., Gaizauskas, R., Huyck, C., Mitchell, B., Cunningham, H., and Wilks, Y. (1998). University of Sheffield: Description of the LaSIE-II system and used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. SAIC.

[Isozaki and Kazawa, 2002] Isozaki, H. and Kazawa, H. (2002). Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th International Conference on Computational Linguistics*, volume 1 of *COLING '02*, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Jia et al., 2009] Jia, H., Newman, J., and Tianfield, H. (2009). A new formal concept analysis based learning approach to ontology building. In Sicilia, M.-A. and Lytras, M., editors, *Metadata and Semantics*, pages 433–444. Springer US.

[Kazama and Torisawa, 2007] Kazama, J. and Torisawa, K. (2007). A new perceptron algorithm for sequence labeling with non-local features. In Eisner, J., editor, *EMNLP-CoNLL*, pages 315–324. ACL.

[Kiryakov et al., 2003] Kiryakov, A., Popov, B., Ognyanoff, D., Manov, D., Kirilov, A., and Goranov, M. (2003). Semantic annotation, indexing, and retrieval. In Fensel, D., Sycara, K., and Mylopoulos, J., editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 484–499. Springer Berlin Heidelberg.

[Klein, 2004] Klein, M. (2004). *Change Management for Distributed Ontologies*. PhD thesis, Vrije Universiteit Amsterdam.

[Kogut and Holmes, 2001] Kogut, P. and Holmes, W. (2001). Aerodaml: Applying information extraction to generate daml annotations from web pages. In *First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation*.

[Kosala and Blockeel, 2000] Kosala, R. and Blockeel, H. (2000). Web mining research: A survey. *SIGKDD Explor. Newsl.*, 2(1):1–15.

[Krötzsch et al., 2007] Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., and Studer, R. (2007). Semantic wikipedia. *Web Semant.*, 5(4):251–261.

[Kuznetsov, 1993] Kuznetsov, S. (1993). A fast algorithm for computing all intersections of objects in a finite semi-lattice. *Automatic documentation and Mathematical linguistics*, 27(5):11–21.

[Kuznetsov, 2004] Kuznetsov, S. (2004). Machine learning and formal concept analysis. In Eklund, P., editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 287–312. Springer Berlin Heidelberg.

[Kuznetsov, 2007] Kuznetsov, S. (2007). On stability of a formal concept. *Annals of Mathematics and Artificial Intelligence*, 49(1-4):101–115.

[Kuznetsov and Makhalova, 2015] Kuznetsov, S. and Makhalova, T. (2015). Concept interestingness measures: a comparative study. In Yahia, S. B. and Konecny, J., editors, *CLA*, volume 1466 of *CEUR Workshop Proceedings*, pages 59–72. CEUR-WS.org.

[Kuznetsov and Obiedkov, 2002] Kuznetsov, S. O. and Obiedkov, S. A. (2002). Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2-3):189–216.

[Liao et al., 2015] Liao, Y., Lezoche, M., Panetto, H., Boudjlida, N., and Rocha Loures, E. (2015). Semantic annotation for knowledge explicitation in a product lifecycle management context: a survey. *Computers in Industry*, 71:24–34.

[Lin, 2008] Lin, Y. (2008). *Semantic Annotation for Process Models: Facilitating Process Knowledge Management via Semantic Interoperability*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway.

[Luong, 2007] Luong, P. (2007). *Gestion de l'évolution d'un Web sémantique d'entreprise*. These, École Nationale Supérieure des Mines de Paris.

[Maedche et al., 2002] Maedche, A., Pekar, V., and Staab, S. (2002). Ontology learning part one - on discovering taxonomic relations from the web. In *Proceedings of the Web Intelligence conference*, pages 301–322. Springer Verlag.

[Maedche and Staab, 2000a] Maedche, A. and Staab, S. (2000a). Discovering conceptual relations from text. In *Proc. of ECAI'2000*, pages 321–325.

[Maedche and Staab, 2000b] Maedche, A. and Staab, S. (2000b). Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, July, 5-7, 2000*. KSI.

[Maedche and Staab, 2001] Maedche, A. and Staab, S. (2001). Ontology learning for the semantic web. *Intelligent Systems, IEEE*, 16(2):72–79.

[Maio et al., 2014] Maio, C., Fenza, G., Gallo, M., Loia, V., and Senatore, S. (2014). Formal and relational concept analysis for fuzzy-based automatic semantic annotation. *Applied Intelligence*, 40(1):154–177.

[Merwe et al., 2004] Merwe, D., Obiedkov, S., and Kourie, D. (2004). Addintent: A new incremental algorithm for constructing concept lattices. In Eklund, P., editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 205–206. Springer, Berlin/Heidelberg.

[Messai et al., 2008] Messai, N., Devignes, M.-D., Napoli, A., and Smail-Tabbone, M. (2008). Extending attribute dependencies for lattice-based querying and navigation. In Eklund, P. and Haemmerlé, O., editors, *Conceptual Structures: Knowledge Visualization and Reasoning*, volume 5113 of *Lecture Notes in Computer Science*, pages 189–202. Springer Berlin Heidelberg.

[Michalski et al., 1998] Michalski, R., Bratko, I., and Bratko, A., editors (1998). *Machine Learning and Data Mining; Methods and Applications*. John Wiley & Sons, Inc., New York, NY, USA.

[Missikoff and Scholl, 1989] Missikoff, M. and Scholl, M. (1989). An algorithm for insertion into a lattice: Application to type classification. In *Foundations of Data Organization and Algorithms*, pages 64–82, New York, NY, USA. Springer-Verlag New York, Inc.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

[Nadeau and Sekine, 2007] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26. Publisher: John Benjamins Publishing Company.

[Napoli, 2005] Napoli, A. (2005). A smooth introduction to symbolic methods for knowledge discovery. Technical report, Inria technical report inria-00001210.

[Ning et al., 2010] Ning, L., Guany, L., and Li, S. (2010). Using formal concept analysis for maritime ontology building. In *Proceedings of the International Forum on Information Technology and Applications*, volume 2, pages 159–162, Washington, DC, USA. IEEE Computer Society.

[Norris, 1978] Norris, E. (1978). An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2):243–250.

[Obitko et al., 2004] Obitko, M., Snásel, V., and Smid, J. (2004). Ontology design with formal concept analysis. In *Proceedings of the International Workshop on Concept Lattices and their Applications (CLA)*, Ostrava, Czech Republic.

[Oren et al., 2006] Oren, E., Moller, K., Scerri, S., Handschuh, S., and Sintek, M. (2006). What are semantic annotations? Technical report, DERI Galway.

[Pawlak, 1992] Pawlak, Z. (1992). *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Norwell, MA, USA.

[Pernelle et al., 2002] Pernelle, N., Rousset, M.-C., Soldano, H., and Ventos, V. (2002). Zoom: a nested galois lattices-based system for conceptual clustering. *Journal of Experimental & Theoretical Artificial Intelligence*, 14:157–187.

[Peroni et al., 2008] Peroni, S., Motta, E., and d'Aquin, M. (2008). Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In Domingue, J. and Anutariya, C., editors, *The Semantic Web*, volume 5367 of *Lecture Notes in Computer Science*, pages 242–256. Springer Berlin Heidelberg.

[Piskova et al., 2014] Piskova, L., Horvath, T., and Krajci, S. (2014). Ranking formal concepts by using matrix factorization. In *Proceedings of the 12th International Conference on Formal Concept Analysis - ICFCA 2014*.

[Poelmans et al., 2013] Poelmans, J., Ignatov, D., Kuznetsov, S., and Dedene, G. (2013). Formal concept analysis in knowledge processing: A survey on applications. *Expert Systems with Applications*, 40(16):6538–6560.

[Ponomareva et al., 2007] Ponomareva, N., Pla, F., Molina, A., and Rosso, P. (2007). Biomedical named entity recognition: A poor knowledge hmm-based approach. In Kedad, Z., Lammari, N., Métais, E., Meziane, F., and Rezgui, Y., editors, *Natural Language Processing and Information Systems*, volume 4592 of *Lecture Notes in Computer Science*, pages 382–387. Springer Berlin Heidelberg.

[Popov et al., 2004] Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., and Kirilov, A. (2004). Kim: a semantic platform for information extraction and retrieval. *Nat. Lang. Eng.*, 10(3-4):375–392.

[Reeve and Han, 2005] Reeve, L. and Han, H. (2005). Survey of semantic annotation platforms. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, SAC '05, pages 1634–1638, New York, NY, USA. ACM.

[Rindflesch and Fiszman, 2003] Rindflesch, T. C. and Fiszman, M. (2003). The interaction of domain knowledge and linguistic structure in natural language processing: Interpreting hypernymic propositions in biomedical text. *Journal of Biomedical Informatics*, 36:462–477.

[Rouane-Hacene et al., 2007] Rouane-Hacene, M., Huchard, M., Napoli, A., and Valtchev, P. (2007). A proposal for combining formal concept analysis and description logics for mining relational data. In Kuznetsov, S. and Schmidt, S., editors, *5th International Conference Formal Concept Analysis (ICFCA)*, volume 4390 of *Lecture Notes in Artificial Intelligence*, pages 51–65, Clermont-Ferrand, France. Springer.

[Rouane-Hacene et al., 2008] Rouane-Hacene, M., Napoli, A., Valtchev, P., Toussaint, Y., and Bendaoud, R. (2008). Ontology learning from text using relational concept analysis. In *Proceedings of the 2008 International MCETECH Conference on e-Technologies*, pages 154–163, Washington, DC, USA. IEEE Computer Society.

[Rouane-Hacene et al., 2011] Rouane-Hacene, M., Valtchev, P., and Nkambou, R. (2011). Supporting ontology design through large-scale fca-based ontology restructuring. In Andrews, S., Polovina, S., Hill, R., and Akhgar, B., editors, *Conceptual Structures for Discovering Knowledge*, volume 6828 of *Lecture Notes in Computer Science*, pages 257–269. Springer Berlin Heidelberg.

[Sarawagi, 2008] Sarawagi, S. (2008). *Information extraction*, volume 1 of *Foundations and trends in databases*. Now Publishers Inc., Hanover, MA, USA.

[Schaffert, 2006] Schaffert, S. (2006). Ikewiki: A semantic wiki for collaborative knowledge management. In *1st International Workshop on Semantic Technologies in Collaborative Applications (STICA)*.

[Shi et al., 2011] Shi, L., Toussaint, Y., Napoli, A., and Blansch, A. (2011). Mining for reengineering: an application to semantic wikis using formal and relational concept analysis. In

Proceedings of the extended semantic web conference on The semanic web: research and applications (ESWC), volume Part II, pages 421–435. Springer-Verlag.

[Soldano and Ventos, 2011] Soldano, H. and Ventos, V. (2011). Abstract concept lattices. In Petko, V. and Robert, J., editors, *Formal Concept Analysis*, volume 6628 of *Lecture Notes in Computer Science*, pages 235–250. Springer Berlin Heidelberg.

[Sowa, 2000] Sowa, J. (2000). Ontology, metadata, and semiotics. In Ganter, B. and Mineau, G., editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues*, volume 1867 of *Lecture Notes in Computer Science*, pages 55–81. Springer Berlin Heidelberg.

[Stojanovic, 2004] Stojanovic, L. (2004). *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe, Germany.

[Studer et al., 1998] Studer, R., Benjamins, V., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197.

[Stumme, 1996] Stumme, G. (1996). Attribute exploration with background implications and exceptions. In Bock, H.-H. and Polasek, W., editors, *Data Analysis and Information Systems*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 457–469. Springer Berlin Heidelberg.

[Stumme, 2002] Stumme, G. (2002). Efficient data mining based on formal concept analysis. In Hameurlain, A., Cicchetti, R., and Traunmüller, R., editors, *Database and Expert Systems Applications*, volume 2453 of *Lecture Notes in Computer Science*, pages 534–546. Springer Berlin Heidelberg.

[Stumme, 2009] Stumme, G. (2009). Formal concept analysis. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 177–199. Springer Berlin Heidelberg.

[Stumme and Maedche, 2001] Stumme, G. and Maedche, A. (2001). Fca-merge: Bottom-up merging of ontologies. In *17th International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 225–234, San Francisco, CA. Morgan Kaufmann Publishers, Inc.

[Szulman et al., 2010] Szulman, S., Charlet, J., Aussenac-Gilles, N., Nazarenko, A., Teguiak, V., and Sardet, E. (2010). Dafoe : an ontology building platform from texts or thesauri. In *Proceedings of International Conference on Knowledge Engineering and Knowledge Management (EKAW)*.

[Talantikite et al., 2009] Talantikite, H., Aissani, D., and Boudjlida, N. (2009). Semantic annotations for web services discovery and composition. *Computer Standards & Interfaces*, 31(6):1108–1117.

[Tallis, 2003] Tallis, M. (2003). Semantic word processing for content authors. In *In Workshop Notes of the Knowledge Markup and Semantic Annotation Workshop (SEMANNOT 2003), Second International Conference on Knowledge Capture (K-CAP 2003)*.

[Tang and Toussaint, 2013] Tang, M. and Toussaint, Y. (2013). A collaborative approach for fca-based knowledge extraction. In *The Tenth International Conference on Concept Lattices and Their Applications - CLA'13*, La Rochelle, France.

[Tartir et al., 2010] Tartir, S., Arpinar, I., and Shethh, A. (2010). Ontological evaluation and validation. In Poli, P., Healy, M., and Kameas, A., editors, *Theory and Applications of Ontology: Computer Applications*, pages 115–130. Springer Netherlands.

[Tissaoui et al., 2011] Tissaoui, A., Aussenac-Gilles, N., Hernandez, N., and Laublet, P. (2011). Evonto - joint evolution of ontologies and semantic annotations. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD)*, pages 226–231, Paris, France.

[Uren et al., 2006] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., and Ciravegna, F. (2006). Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semant.*, 4(1):14–28.

[Valtchev and Missaoui, 2001] Valtchev, P. and Missaoui, R. (2001). Building galois (concept) lattices from parts: Generalizing the incremental approach. In Delugach, H. and Stumme, G., editors, *Proceedings of the international conference on Conceptual Structures: Knowledge Visualization and Reasoning (ICCS)*, volume 2120 of *Lecture Notes in Computer Science*, pages 290–303. Springer Verlag.

[Vargas-Vera et al., 2002] Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., and Ciravegna, F. (2002). Mnm: Ontology driven semi-automatic and automatic support for semantic markup. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, EKAW '02, pages 379–391, London, UK. Springer-Verlag.

[Wille, 1989] Wille, R. (1989). Knowledge acquisition by methods of formal concept analysis. In *Proceedings of the Conference on Data Analysis, Learning Symbolic and Numeric Knowledge*, pages 365–380, Commack, NY, USA. Nova Science Publishers, Inc.

[Wille, 2002] Wille, R. (2002). Why can concept lattices support knowledge discovery in databases? *Journal of Experimental and Theoretical Artificial Intelligence*, 14(2-3):81–92.

[Wong et al., 2012] Wong, W., Liu, W., and Bennamoun, M. (2012). Ontology learning from text: A look back and into the future. *ACM Computing Surveys*, 44(4):20:1–20:36.

[Zablith et al., 2013] Zablith, F., Antoniou, G., d' Aquin, M., Flouris, G., Kondylakis, H., Motta, E., Plexousakis, D., and Sabou, M. (2013). Ontology evolution: a process-centric survey. *The Knowledge Engineering Review*, FirstView:1–31.

[Zhang et al., 2010] Zhang, H., Li, Y.-F., and Tan, H. (2010). Measuring design complexity of semantic web ontologies. *The Journal of Systems and Software*, 83(5):803–814.

[Zhou and J.Su, 2004] Zhou, G. and J.Su (2004). Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, JNLPBA '04, pages 96–99, Stroudsburg, PA, USA. Association for Computational Linguistics.

Résumé

Dans cette thèse, nous présentons notre méthodologie de la connaissance interactive et itérative pour une extraction des textes - le système KESAM: Un outil pour l'extraction des connaissances et le Management de l'Annotation Sémantique. Le KESAM est basé sur l'analyse formelle du concept pour l'extraction des connaissances à partir de ressources textuelles qui prend en charge l'interaction aux experts. Dans le système KESAM, l'extraction des connaissances et l'annotation sémantique sont unifiées en un seul processus pour bénéficier à la fois l'extraction des connaissances et l'annotation sémantique. L'annotations sémantiques sont utilisées de formaliser la source de la connaissance dans les textes et de garder la traçabilité entre le modèle de la connaissance et la source de la connaissance. Le modèle de connaissance est, en revanche, utilisé d'améliorer les annotations sémantiques. Le processus KESAM a été conçu pour préserver en permanence le lien entre les ressources (textes et annotations sémantiques) et le modèle de la connaissance. Le noyau du processus est l'Analyse Formelle de Concepts (AFC) qui construit le modèle de la connaissance, i.e. le treillis de concepts, et assure le lien entre le modèle et les annotations des connaissances. Afin d'obtenir le résultat du treillis aussi près que possible aux besoins des experts de ce domaine, nous introduisons un processus itératif qui permet une interaction des experts sur le treillis. Les experts sont invités à évaluer et à affiner le réseau; ils peuvent faire des changements dans le treillis jusqu'à ce qu'ils parviennent à un accord entre le modèle et leurs propres connaissances ou le besoin de l'application. Grâce au lien entre le modèle des connaissances et des annotations sémantiques, le modèle de la connaissance et les annotations sémantiques peuvent co-évoluer afin d'améliorer leur qualité par rapport aux exigences des experts du domaine. En outre, à l'aide de l'AFC de la construction des concepts avec les définitions des ensembles des objets et des ensembles d'attributs, le système KESAM est capable de prendre en compte les deux concepts atomiques et définis, à savoir les concepts qui sont définis par un ensemble des attributs.

Afin de combler l'écart possible entre le modèle de représentation basé sur un treillis de concept et le modèle de représentation d'un expert du domaine, nous présentons ensuite une méthode formelle pour l'intégration des connaissances d'expert en treillis des concepts d'une manière telle que nous pouvons maintenir la structure des concepts du treillis. La connaissance d'expert est codé comme un ensemble de dépendance de l'attribut qui est aligné avec l'ensemble des implications fournies par le concept du treillis, ce qui conduit à des modifications dans le treillis d'origine. La méthode permet également aux experts de garder une trace des changements qui se produisent dans le treillis d'origine et la version finale contrainte, et d'accéder à la façon dont les concepts dans la pratique sont liés à des concepts émis automatiquement à partir des données. Nous pouvons construire les treillis contraints sans changer les données et fournir la trace des changements en utilisant des projections extensives sur treillis. À partir d'un treillis d'origine, deux projections différentes produisent deux treillis contraints différents, et, par conséquent, l'écart entre le modèle de représentation basée sur un treillis de réflexion et le modèle de représentation d'un expert du domaine est rempli avec des projections.

Mots-clés: Analyse formelle de concepts, extraction de connaissances, annotation semantique, implication de l'attribut, dépendance de l'attribut

Abstract

In this thesis, we present a methodology for interactive and iterative extracting knowledge from texts - the KESAM system: A tool for Knowledge Extraction and Semantic Annotation Management. KESAM is based on Formal Concept Analysis for extracting knowledge from textual resources that supports expert interaction. In the KESAM system, knowledge extraction and semantic annotation are unified into one single process to benefit both knowledge extraction and semantic annotation. Semantic annotations are used for formalizing the source of knowledge in texts and keeping the traceability between the knowledge model and the source of knowledge. The knowledge model is, in return, used for improving semantic annotations. The KESAM process has been designed to permanently preserve the link between the resources (texts and semantic annotations) and the knowledge model. The core of the process is Formal Concept Analysis that builds the knowledge model, i.e. the concept lattice, and ensures the link between the knowledge model and annotations. In order to get the resulting lattice as close as possible to domain experts' requirements, we introduce an iterative process that enables expert interaction on the lattice. Experts are invited to evaluate and refine the lattice; they can make changes in the lattice until they reach an agreement between the model and their own knowledge or application's need. Thanks to the link between the knowledge model and semantic annotations, the knowledge model and semantic annotations can co-evolve in order to improve their quality with respect to domain experts' requirements. Moreover, by using FCA to build concepts with definitions of sets of objects and sets of attributes, the KESAM system is able to take into account both atomic and defined concepts, i.e. concepts that are defined by a set of attributes.

In order to bridge the possible gap between the representation model based on a concept lattice and the representation model of a domain expert, we then introduce a formal method for integrating expert knowledge into concept lattices in such a way that we can maintain the lattice structure. The expert knowledge is encoded as a set of attribute dependencies which is aligned with the set of implications provided by the concept lattice, leading to modifications in the original lattice. The method also allows the experts to keep a trace of changes occurring in the original lattice and the final constrained version, and to access how concepts in practice are related to concepts automatically issued from data. The method uses extensional projections to build the constrained lattices without changing the original data and provide the trace of changes. From an original lattice, two different projections produce two different constrained lattices, and thus, the gap between the representation model based on a concept lattice and the representation model of a domain expert is filled with projections.

Keywords: Formal concept analysis, knowledge extraction, semantic annotation, attribute implication, attribute dependency